Commodore WORLD

Nº 2. MARZO 1984

275 PTAS.

VIAJES a ESTADOS UNIDOS
Para nuestros lectores
Página 4





GALERIA DE SOFT



El programa para el procesador de textos para el Commodore-64. EASY SCRIPT convierte al C-64 en un procesador de textos muy potente que dispone de todas las facilidades de una máquina dedicada que pudiera costar **miles de libras**.

Está escrito totalmente en lenguaje máquina, el cual permite una velocidad realmente alta, con una capacidad de aproximadamente 30.000 caracteres de texto, es decir, más de 750 líneas de texto. Con EASY SCRIPT los textos se pueden crear, salvar, recuperar y modificar mediante el uso bien, de la unidad de disco 1541 o la grabadora C2N. Incluso se puede transferir la información tanto de la unidad de disco a la grabadora como de la grabadora a la unidad de disco.

El texto se almacena en disco o en cinta en unos ficheros de datos estandard que pueden ser leídos por los programas en Basic. De esta forma EASY SCRIPT dispone de la capacidad de leer la información creada a partir de un programa en Basic.



DE COMPRAS

PRECIO: 20.000 PTS.

Para introducir un texto muchas teclas funcionan igual que para un programa en Basic, por ejemplo las teclas del control del cursor.

Ademmás, 55 potentes comandos de editar están disponibles al pulsar una tecla.

Una vez introducido el texto, existen más de 35 comandos de formateado que pueden ser utilizados para modificar la forma en que se imprime el texto.

Algunas de las facilidades disponibles en EASY SCRIPT se presentan a continuación:

- Búsqueda y Sustitución.
- Relleno automático de bloques variables.
- Centrado y Justificación.
- Modalidades de Borrado e Inserción.
- Desplazamiento (Scrolling) en todas las direcciones.
- Series de ficheros encadenados.
- Tabulaciones horizontales y verticales.
- Modalidad y encolumnado de números decimales.
- Cabeceras y Pies.

EASY SCRIPT

Procesador de Textos para C-64

EASY SCRIPT no solamente funciona con las impresoras matriciales 1515 y 1525, sino que también maneja otras marcas populares, incluyendo la serie EPSON MX y hasta las impresoras de margarita con letras de calidad de imprenta.

El texto se introduce fácilmente en EASY SCRIPT y sólo hace falta pulsar la tecla RETURN al final de cada párrafo.

- Numeración automática de las páginas.
- Manejo de discos y cintas.
- Combinación (merge) de párrafos o ficheros.
- Pleno control de la impresora.

En el próximo número se publicará un amplio artículo sobre cómo generar Caracteres Especiales con EASY SCRIPT

LAS TORRES DE HANOI

Para el VIC-20 — **PRECIO: 1.500 PTS.**

Tradicionalmente este juego se hace con tres palos verticales y unos aros de diámetros diferentes. Al inicio se colocan todos los aros en un mismo palo de menor a mayor. El juego consiste en pasarlos a otro palo. Solo hay dos reglas: Nunca puede estar un aro sobre otro de diámetro menor, y sólo se puede tomar el aro superior de cada torre.

Estos paquetes están distribuidos por Microelectrónica y Control

MATEMATICAS-1

PROGRAMA NIVEL BUP PARA VIC-20 CON 8 ó 16 K

PRECIO: 2.000 PTS. en cinta

Este es un programa que plantea tests de matemáticas y geometría. El VIC preguntará el tema que prefiere, Usted cargará la cassette de datos, una vez el VIC haya encontrado el cuestionario, usted podrá empezar a contestar. Usted puede pasar pulsando"Return" o finalizar el test tecleando "Fin". Una vez finalizado el test, se le presentará la puntuación, el tiempo empleado y la posibilidad de contestar las preguntas pasadas. Si dice "N", el VIC ofrecerá un menú con tres posibilidades:

—Otro test
—Revisar test
—Fin

En cuyo caso el VIC presentará la portada del programa.

NOTA: En el test de tiempo no dice si la contestación es correcta o no.



SUMARIO 2

Marzo 1984

CONTENIDO

PROXIMO NUMERO

- Curso lenguaje Máquina
- Exprimiendo el jugo a las teclas de funciones
- Base de datos para el VIC 20 y el C-64
- Caracteres especiales con Easy Script
- Atajo hacia el color
- Chavales... ¡Sorpresa!
- Recetario de Pokes
- Juegos
- Club Commodore

y todas vuestras colaboraciones

¡OS ESPERAMOS!

Commodore World es publicado en colaboración entre Microelectrónica y Control-Commodore y SIMSA

EQUIPO

Manuel AMADO; Adela LOPEZ; María LOPEZ; Juan MARTINEZ;
Pere MASATS; Jeffrey MILLS; Rafael NAVARRO; Fernando M. RODRIGUEZ;
Diego ROMERO; Albert SANGLAS; Manuel SANS; Jordi SASTRE; Valerie SHANKS...
... Y NUESTROS LECTORES

SIMSA
Coordinador María López
Pedro Muguruza, 4-8ºB — Madrid-16 — Tlf.: (91) 259 54 78

Sant Gervasi de Cassoles, 39 despacho 4
Barcelona-22 — Tlf.: (93) 212 73 45

MICROELECTRONICA
Y CONTROL-COMMODORE
Coordinador Pere Masats
Taquígrafo Serra, 7-5º
Barcelona 29
Teléfono (93) 250 51 03/02

Imprime: IBERDOS, S.A. Germán Pérez Carrasco, 24. Madrid-27
Depósito Legal: M-2944-1984

SUPER-INTERESANTISIMO

Un lector... (¿o más...?) viajará a Estados Unidos con Commodore World

Porque estábamos seguros del éxito de Commodore World, pero no tanto, sabíamos que iba a "caer bien" y sobre todo que iba a llenar un hueco muy impotante en la gran familia Commodore —sin embargo, sinceramente, no nos esperábamos la súper gran acogida que ha tenido—. Hoy al redactar estas líneas, 15 de febrero, a sólo 15 días de su salida a la calle, hemos superado la línea de los 3.500 suscriptores y nos estamos acercando rápidamente a los 4.000 y el buzón se nos ha quedado pequeño recibiendo correspondencia de todos los que formáis el gran equipo de Commodore World. Y... se acabó el hablar y darnos auto-bombo, vamos al grano y seguid leyendo, que lo que sigue es de "super-interés" para todos.

Suscriptores y el nº 5.000... ¡BINGO!

Las suscripciones se acumulan en la redacción y en el teléfono. En el momento en que se abre un sobre, se recibe una tarjeta o se apunta una suscripción telefónica, se apunta el número correspondiente a esa suscripción... Pues bien, **el suscriptor al que le corresponda el nº 5.000**, y ya hemos superado los 3.500, **se encontrará automáticamente con un viaje a Estados Unidos, donde visitará la fabrica y casa madre de Commodore...** Sí, habéis leído bien... no es ningún error de imprenta. ¡Bueno! ¿Y los demás suscriptores, qué? —Eso no es justo— vale, de acuerdo, sobre todo para los 4.999 primeros que han hecho posible que el número 5.000 cante BINGO. Pues, aunque no está decidido todavía qué será, adelantaremos que organizaremos algún tinglado para todos, tanto los pre-5.000 como los post-5.000 y **algo especial** para los primeros 4.999 que anunciaremos en cuanto se haya decidido, y veréis que nosotros no nos dormimos en los laureles y hacemos las cosas rápido (adelantamos que hay viajes y aventuras por medio).

Los "VIEJOS LEALES" y los nº 13, 14 y 15

Hemos tenido algunas llamadas de antiguos suscriptores del Club Commodore que no han recibido los nº 13, 14 y 15. Lo más probable es que por coincidir con la época de transición haya habido algún traspapeleo involuntario. A los que os haya ocurrido esto, **escribid** (no por teléfono) a Microelectrónica y Control de Barcelona, c/ Taquígrafo Serra, 7, especificando a la atención de Pere Masats. Pere, tan imprescindible hoy en Commodore World como lo fue antes en Club Commodore, estará encantado de remitiros los ejemplares que os faltan a vuelta de correo.

Market-Club
Mercadillo-Clubs-Bolsa de trabajo

Pretendemos que MARKET-CLUB sea una importante sección al servicio de toda la familia Commodore. A partir de hoy mismo se dividirá en tres apartados: MERCADILLO, CLUBS y, muy importante, BOLSA DE TRABAJO.

MERCADILLO: Será para todo el cambalache, equipos usados, ofertas, demandas, etc.

CLUBS: La propia palabra lo explica. Información de Club Commodore por todo el pais tanto de los que existen como de usuarios que estan buscando uno ya formado en su zona o de aquellos que quieran formar uno nuevo.

BOLSA DE TRABAJO: Este es el apartado que con mayor ilusión comenzamos y pretendemos que sea una importantísima "estrella" de Commodore World. Por un lado, un gran número de "Commodorianos" son gente joven que esta buscando empleo por primera vez, por otro lado, dentro de los "Commodorianos" maduros y semi-maduros hay los que se encuentran en paro y creemos que nuestra revista puede prestar un gran servicio en este campo.

Todos los anuncios seran gratuitos excepto para las Compañías que ofrezcan servicios, productos o empleo a quienes se les cobrará 300 pesetas por línea.

Todos los anuncios llevarán un número de referencia. Os rogamos que cuando ya no se necesite publicar, nos lo comuniquéis, nombrando este número, para suprimirlo.

Programas y errores...

A partir de este número que tenéis en la mano, podemos garantizaros que todos los programas y juegos que se incluyen, vayan completamente comprobados y sin el más mínimo error (dentro de lo humanamente posible). Diego, el "cerebro" del equipo editorial ha hecho todas las horas extraordinarias del mundo para asegurarse de que todos los listados van perfectos.

Sin embargo, lamentamos que en nuestro primer número, debido al gran trabajo de lanzamiento, no pudimos ser tan concienzudos y parece ser que se nos introdujo un duende travieso en la linea 850 de Video Casino-Tiro al Blanco. Un lector, nos acaba de llamar diciendo que tenía problemas con esta línea. Vamos a revisar este programa y os diremos lo que hay en nuestro próximo número. Quisiéramos poder sacaros de dudas ahora mismo pero tenemos que entrar en imprenta y no da tiempo.

...Y servicio de cintas para los más vagos.

Aquellos lectores que os canséis de teclear un programa (no os preocupeis ni os dé vergüenza, ¡sucede en todo el mundo!) (¡por eso existe el soft!) podéis pedir a la editorial la cinta del programa y número de Commodore World en que aparece, adjuntando un cheque por el importe: 875 pesetas por cinta y 75 pesetas para gastos de envío —cuidado, 75 pesetas por paquete, no por cinta—. Los que vivís en Madrid podéis recoger las cintas en la editorial y no pagar las 75 pesetas (llamad antes por teléfono para tenerlas preparadas).

Carta Blanca y Seamos Preguntones

Dos nuevas secciones: CARTA BLANCA habla por sí misma —cartas del lector con todo lo que tengáis que decir, opinar o sugerir— tenéis "carta blanca". SEAMOS PREGUNTONES —seámoslo, por favor— preguntad todo lo que os interese sobre los Commodore, su funcionamiento, formas de darle mayor rendimiento o capacidad, programación, etc. Contestaremos todo lo que podamos y si hay algo que no sepamos contestar, la pregunta queda hecha como desafío a los lectores "listos".

Chavales Especial para vosotros

No nos hemos olvidado de vosotros. Aunque sabemos que los "Commodorianos" infantiles son unos superlistos que pueden manejar un montón de las cosas que aparecen en la revista, a partir del próximo número tendréis la sección especial para los

más jovencillos de la familia, aunque mucho nos tememos que los adultos la van a echar más de un vistazo a hurtadillas. Así que atención, nuestros "duendes" ROM y RAM os esperan en el próximo número y con algún programilla especial para vosotros —que no pase como en "el tren" que luego es papá el primero que lo usa... (¡o mamá!).

Curiosidades sobre los Commodore

El otro día nos dijeron que un famoso campanario de un pueblo está funcionando de nuevo después de muchos años de estar casi callado. El milagro se debe a que esta siendo manejado por un VIC y unos dicen que consigue un cambio de melodías precioso. Los que sepáis de casos en que se emplean los Commodore para cosas que os sorprendan, contádnoslo —puede dar ideas a la gente de lo mucho que pueden hacer con una "maquinita" y al mismo tiempo puede haber anécdotas divertidas.

Colaboraciones

Recordamos que para todas las colaboraciones que se envíen va a haber sorteos y premios (ver Commodore World nº 1). Como los premios especiales a los mejores van a darse por grupos de edad, no olvidéis poner vuestra edad. En nuestro número 1 decíamos, por error de imprenta, que los programas deberían venir todos con cinta o disco —rectificamos— deben venir todos con el listado completo en impresora —la cinta si es posible enviarla, mejor, la devolveremos inmediatamente con otra de regalo. Enviad todas vuestras colaboraciones a nuestra editorial en Madrid.

IMPORTANTE: Recordamos que si no especificáis que por razones personales no queréis que se publique vuestra dirección, todos los colaboradores aparecerán con nombre y dirección para que podáis poneros en contacto unos con otros.

Distribuidores: nota especial para vosotros

Veréis en CARTA BLANCA (página 45) que muchos lectores piden información sobre hard, soft, juegos, etc. Enviadnos listas e información **siempre con precios**. En Galería de Soft (páginas 2 y 59) pretendemos publicar reseñas de paquetes —enviadnos una copia de los programas que deseéis se reseñen para su comprobación. La editorial se reserva el derecho de decisión sobre la publicación de las reseñas. Siempre se publicará el nombre del distribuidor correspondiente.

Suscripciones por teléfono

Podéis suscribiros por teléfono (91) 259 54 78 preguntando por Valerie o María.

Anuncios en la revista

PROVEEDORES: anunciar vuestros productos compatibles con micros Commodore (software, periféricos, interfaces).
TIENDAS DE MICROS Y PRODUCTOS: Anunciar vuestros locales para que nuestros lectores puedan saber a dónde dirigirse.

Llamando a Barcelona: 212 73 45 y preguntar por Neus.

Llamando a Madrid: 259 54 78 y preguntar por Enrique.

Copias de nuestro predecesor Club Commodore

Nosotros no disponemos de copias originales de Club Commodore ya que Microelectrónica y Control las tiene agotadas. Por eso ofrecemos un servicio de fotocopias. Los que deseéis estas fotocopias, enviad por favor el boletín de la página 5. No hay ningún problema en realizar suscripciones por teléfono, sin embargo, el servicio de fotocopias realizarlo, por favor, por escrito. Unos quieren toda la colección, otros desean números sueltos y por escrito no puede haber malos entendidos. Dadnos un margen de dos semanas para su entrega.

EJEMPLARES ATRASADOS DE «CLUB COMMODORE»

Para poder satisfacer la creciente demanda de números atrasados de nuestra Revista, agotada en todas sus ediciones, hemos puesto en marcha un Servicio para suministrar fotocopias de los ejemplares que nos sean solicitados. Para recibir las fotocopias de una o de varias ediciones, no hay más que enviarnos el boletín con los datos indicados.

SERVICIO DE FOTOCOPIAS.— NUMERO DE LA EDICION SOLICITADA.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

(Poner una X debajo del número de edición pedido)

Peticionario: D. ....

Calle .... nº ....

Población .... D.P. .... Provincia ....

Forma de pago por cheque **Precio de la edición fotocopiada: 250 ptas.**

La colección completa del 0 al 15: 2.500 ptas. + 100 ptas. por gastos de envío

Incluyo cheque por .... ptas.

BOLETIN DE SUSCRIPCION — commodore world

NOMBRE .... EDAD ....

DIRECCION ....

POBLACION .... (..........) PROVINCIA ....

TELEF. .... MARCA Y MODELO DEL ORDENADOR ....

CIUDAD DONDE LO COMPRO .... DISTRIBUIDOR ....

APLICACIONES A LAS QUE PIENSA DESTINAR EL EQUIPO ....

....

Deseo iniciar la suscripción con el nº ☐

Adjunto cheque de 2.200 pesetas ☐

Reembolso más gastos del mismo al recibir el primer nº de la suscripción ☐

(Enviar a la dirección del dorso)

Firma,

Extranjero $35. Solamente por correo aéreo

DESEO SUSCRIBIRME A **COMMODORE WORLD** POR UN AÑO AL PRECIO DE 2.200 PTS. DICHA SUSCRIPCION ME DA DERECHO, NO SOLO A RECIBIR LA REVISTA (ONCE NUMEROS ANUALES) SINO A PARTICIPAR EN LAS ACTIVIDADES QUE SE ORGANICEN EN TORNO A ELLA Y QUE PUEDEN SER COORDINACION DE CURSOS DE BASIC, INTERCAMBIOS DE PROGRAMAS, CONCURSOS, ETC.

Clave para interpretar los listados

Todos los listados que se publican en esta Revista han sido ejecutados en el modelo correspondiente de la gama de ordenadores COMMODORE. Para facilitar la edición de los mismos en la Revista y para mejorar su legibilidad por parte del usuario, se les ha sometido a ciertas modificaciones mediante un programa escrito especialmente para ello. Para los programas destinados a los ordenadores VIC-20 y COMMODORE 64, en los que se usan frecuentemente las posibilidades gráficas del teclado, se han sustituido los símbolos gráficos que aparecen normalmente en los listados por una serie de letras entre corchetes [] que indican la secuencia de teclas que se deben pulsar para obtener el carácter deseado. A continuación se da una tabla para aclarar la interpretación de las indicaciones entre corchetes:

[CRSRD] = Tecla cursor hacia abajo (sin SHIFT)

[CRSRU] = Tecla cursor hacia arriba (con SHIFT)

[CRSRR] = Tecla cursor a la derecha (sin SHIFT)

[CRSRL] = Tecla cursor a la izquierda (con SHIFT)

[HOME] = Tecla CLR/HOME (sin SHIFT)

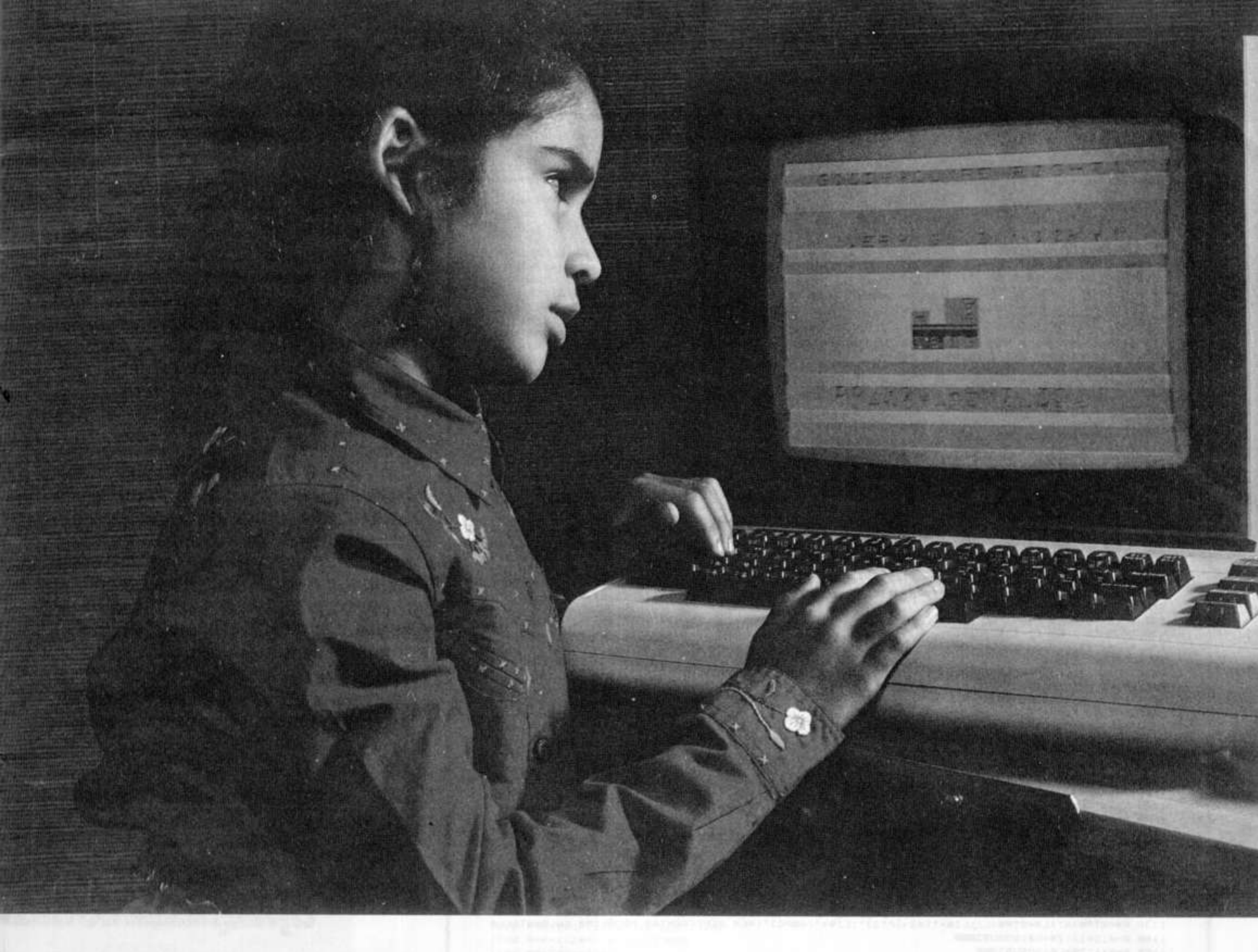
[CLR] = Tecla CLR/HOME (con SHIFT)

Las indicaciones [BLK] a [YEL] corresponden a la pulsación de las teclas de 1 a 8 junto a la tecla CTRL. Lo mismo sucede con [RVSON] y [RVSOF] respecto a la tecla CTRL y las teclas 9 y 10.

El resto de las indicaciones constan de la parte COMM o SHIF seguidas de una letra, número o símbolo —por ejemplo [COMM+] o [SHIFA]—. Esto indica que para obtener el gráfico necesario en el programa deben pulsarse simultáneamente las teclas COMMODORE (la que lleva el logotipo) o una de SHIFT y la tecla indicada por la letra, el número o el símbolo, en el ejemplo anterior: COMMODORE y + o SHIFT y A, respectivamente.

En los signos gráficos además se cuenta el número de veces que aparece. Por ejemplo, [7 CRSRR] equivale a 7 pulsaciones de la tecla cursor a la derecha y [3 SPC] tres pulsaciones de la barra espaciadora. ■

INDICE ANUNCIANTES



"Mathquiz"

Diviértete con las matemáticas

Esto es un programa sencillo que utiliza las características del VIC-20 ó el Commodore 64 (gráficos, colores, cronometraje y sonido) para servir de repaso de las matemáticas estudiadas en la escuela.

COMMODORE 64
o
VIC-20
(sin ampliación)

La rutina en "Mathquiz", un programa escrito para ejecutarse en el VIC-20 no ampliado, es una forma modernizada de las antiguas tarjetas que comprobaban la habilidad del niño para sumar, restar o multiplicar. Dado que el programa presenta no sólo problemas cuyas respuestas normalmente se memorizan, sino también problemas que tienen que ser calculados, se amplían y se refuerzan las técnicas matemáticas aprendidas en los primeros años de la escuela.

Listado 1

```
        MATHQUIZ     VIC 20

1 REM MATHQUIZ PARA EL VIC
2 CL=6:REM COLOR DE LOS NUMEROS
5 DEF FNL(X)=X+22*R+C-PP
10 S2=36875:V=S2+3:S4=S2+2
30 GOTO900
99 REM CANCION
100 POKEV,15
102 IFCA-6<=0THEN210:REM LA CANTIDAD DE CANCION TOCADA DEPENDE DE LA PUNTUACION
103 REM SI LA NOTA ES -1 ENTONCES PARAR
105 READ P:IF P=-1 THEN202
120 READ D:POKE S2,P
160 FOR N=1TO150*D:NEXTN:REM CAMBIAR 150 PARA ALTERAR EL RITMO
170 POKE S2,0:FORN=1TO20:NEXTN
190 GOTO105
202 CA=CA-1:GOTO102
210 POKE V,0:RESTORE:GOTO1600
299 REM TONO,RITMO
300 DATA183,2,195,1,215,3,209,2,195,1,187,3
310 DATA183,2,183,1,183,1,187,1,195,1,201,3,195,3,-1
320 DATA183,2,195,1,215,3,209,2,195,1
340 DATA187,3,183,2,195,1,195,1,201,1,207,1,209,3,209,3,-1
350 DATA215,1.5,195,0.5,195,1,207,1,201,1,195,1,183,2
360 DATA195,1,209,3,201,2,209,1,215,2,209,1
370 DATA207,3,195,3,-1,183,2,195,1,215,3,209,2,195,1
380 DATA187,3,183,2,195,1,195,1,201,1,207,1,209,3,209,3,-1
900 PRINT"[CLR]":PRINT"[SPC]SUMA=S":PRINT"[SPC]RESTA=R"
905 PRINT"[SPC]MULTIPLICACION=M":PRINT"[SPC]ELECCION[SPC]?"
910 GETL$:IFL$=""THEN910
920 IFL$="R"THEN CH=-1:GOTO1000
930 IFL$="S"THEN CH=1:GOTO1000
940 IFL$="M"THEN CH=0:GOTO1000
950 GOTO900
1000 PRINT"[CLR]":PRINT"NIVEL[SPC]DE[SPC]DIFICULTAD?":PRINT"1,2,O[SPC]3;":PRINT"
(1[SPC]ES[SPC]MAS[SPC]FACIL)"
1005 PRINT"ELECCION[SPC]?"
1010 GETL$:IFL$=""THEN1010
1020 IFL$<"1"ORL$>"3"THEN1010
1030 F=10↑(VAL(L$)-1)
1060 CA=0:FORII=1TO10:PRINT"[CLR]"
1070 K=INT(RND(1)*F*10):REM ELEGIR EL PROBLEMA
1074 F1=F
1075 IFCH=0ANDF>1THENF1=F/10
1080 L=INT(RND(1)*F1*10)
1090 ON CH+1GOTO1110,1120
1100 SN=45:IFL>KTHEN1070:REM ELIMINA RESTAS CON RESULTADO NEGATIVO
1105 ANS=K-L:GOTO1130
1110 SN=24:IF INT(L/10)=0 OR INT(L/10)=L/10 THEN 1118
1112 GOTO1080
1118 ANS=K*L:IFINT(ANS/10000)>0THEN1080
1119 GOTO1130
1120 SN=43:ANS=K+L
1130 K$=STR$(K):L$=STR$(L):LI$="[4SHIFTD]":LJ$="[4SPC]":REM REPRESENTAR PROBLEMA EN PANTALLA
1140 R=6:C=11:Z$=K$:GOSUB3000
1150 R=R+1:Z$=L$:GOSUB3000
1160 C=C-LEN(L$)-1:PP=0:POKEFNL(7680),SN:POKEFNL(38400),CL
1170 R=R+1:C=11:Z$=LI$:GOSUB3000
1180 R=R+1:Z$=LJ$:GOSUB3000
1190 MM=7680+22*R+C
1200 Z1=INT(TI/100)
1210 GOSUB2210
1220 IFZIP=1THEN2000:REMTIEMPO AGOTADO
1230 POKEMM,ASC(AZ$):MM=MM-1:AP=VAL(AZ$)
1250 IFLEN(STR$(ANS))<3THEN1440
1260 GOSUB2210:IFZIP=1THEN2000
1265 IFZAP=1THEN1210
1270 POKEMM,ASC(AZ$):MM=MM-1:AP=AP+10*VAL(AZ$)
1280 IFLEN(STR$(ANS))<4THEN1440
1290 GOSUB2210:IFZIP=1THEN2000
1295 IFZAP=1THEN1260
1300 POKEMM,ASC(AZ$):MM=MM-1:AP=AP+100*VAL(AZ$)
1310 IFLEN(STR$(ANS))<5THEN1440
1320 GOSUB2210:IFZIP=1THEN2000
1325 IFZAP=1THEN1290
1330 POKEMM,ASC(AZ$):AP=AP+1000*VAL(AZ$)
1440 IFAP=ANSTHENPRINT"[2SPC]BIEN-HAS[SPC]ACERTADO":CA=CA+1:GOTO1490
1450 PRINT"[2SPC]NO-LA[SPC]RESPUESTA[SPC]ES":PRINTANS
1490 FORDL=1TO750:GETL$:NEXTDL,II
1500 PRINT"[CLR]":PRINT"[SPC]HAS[SPC]ACERTADO":PRINT"[SPC]";CA;"[SPC]DE[SPC]DIEZ (SPC]VECES"
1505 IFCA<4THENCA=4
1510 ONCA-4GOTO1530,1540,1550,1560,1570,1580
1520 PRINT:PRINT"ASEGURATE[SPC]QUE[SPC]COLOQUES";"[2SPC]EL[SPC]NUMERO[SPC]DONDE[SPC]SE"
1525 PRINT"[SPC]ENCUENTRA[SPC]EL[SPC]SIMBOLO[6SPC]DESTELLANTE":GOTO1600
1530 PRINT:PRINT"[SPC]OK--":PRINT"[SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO1600
1540 PRINT:PRINT"[SPC]BIEN--":PRINT"[2SPC]INTENTA[SPC]UNOS[SPC]MAS!!":JOTO1600
1550 PRINT:PRINT"MUY[SPC]BIEN--":PRINT"[SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO100
1560 PRINT:PRINT"[SPC]EXCELENTE--":PRINT"[SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO100
1570 PRINT:PRINT"[SPC]FABULOSO--":PRINT"INTENTA[SPC]UNOS[SPC]MAS!!":GOTO100
1580 PRINT:PRINT"[SPC]FANTASTICO!!!--":PRINT"INTENTA[SPC]UNOS[SPC]MAS!!":GOTO100
1600 PRINT:PRINT"QUIERES[SPC]MAS?[SPC](SI[SPC]O[SPC]NO)"
1610 GETD$:IFD$=""THEN1610
1620 IFD$="S"THEN900
1630 END
2000 PRINT"[SPC]TIEMPO[SPC]AGOTADO":PRINT"LA[SPC]RESPUESTA[SPC]ERA";ANS
2010 POKES4,250:POKEV,15:FORJJ=1TO50:NEXTJJ:POKES4,0:POKEV,0:GOTO1490
2210 ZIP=0:ZAP=0:GETAZ$:IFINT(TI/100)>Z1+5*FTHEN2250
2215 KK=KK+1:IFKK/2=INT(KK/2)THENPOKEMM,70:GOTO2220
2217 POKEMM,102
2220 IFAZ$=""THEN2210
2222 IFASC(AZ$)=136THENCL=CL+1:REM CAMBIA EL COLOR
2223 IFCL<2ORCL>8THENCL=2
2224 REM BORRAR
2225 IFASC(AZ$)=29THENMM=MM+1:AP=AP-10↑(7889-MM)*(PEEK(MM)-48):ZAP=1:RETURN
2230 IFASC(AZ$)<48 OR ASC(AZ$)>57THEN2210
2240 RETURN
2250 ZIP=1:RETURN
3000 FORPP=0TOLEN(Z$)-1:POKEFNL(7680),ASC(MID$(Z$,LEN(Z$)-PP,1))
3010 POKEFNL(38400),CL:NEXTPP:RETURN
```

Existen tres niveles de dificultad para cada operación matemática (1, 2 ó 3), siendo el nivel 1 el más fácil. Estos niveles corresponden a la suma o resta de números de uno, dos o tres dígitos. Para la multiplicación, los niveles corresponden al producto de dos números de un solo dígito, un número de dos dígitos por un número de un solo dígito, o un número de tres dígitos por un número de dos dígitos divisible por diez.

El programa selecciona cada uno de los dos números según el nivel de dificultad elegido, pero no permite que se dé un número negativo en una operación de resta. Si la respuesta dada por el estudiante no es correcta, el programa presenta una nota y la solución correcta. El estudiante dispone de un tiempo fijo para solucionar cada problema, y se permite más tiempo para los problemas más difíciles. Si el tiempo se agota, un sonido

(beep) le indica al estudiante que mire la pantalla, donde se representa la solución correcta.

El sonido de la música

Cuando se han estudiado diez problemas, la puntuación del estudiante aparece en pantalla, junto con un premio musical. La canción que aparece en el programa es la preferida de mi hija, Kathy, pero suele ser sustituida por cualquier canción. Esta parte del programa (las líneas 100-210) ha sido sacada del manual del VIC-20. La canción (las líneas 300-380) se divide en cuatro partes (separadas mediante un -1), y se toca toda la canción cuando se da una puntuación perfecta. Para cada pregunta no acertada, la canción dura menos, proporcionando así un incentivo para que todas las respuestas sean correctas.

Cuando se ejecuta, el programa pide una operación matemática (suma, resta o multiplicación), seguida por el

tan divertido
n el "mathquiz"

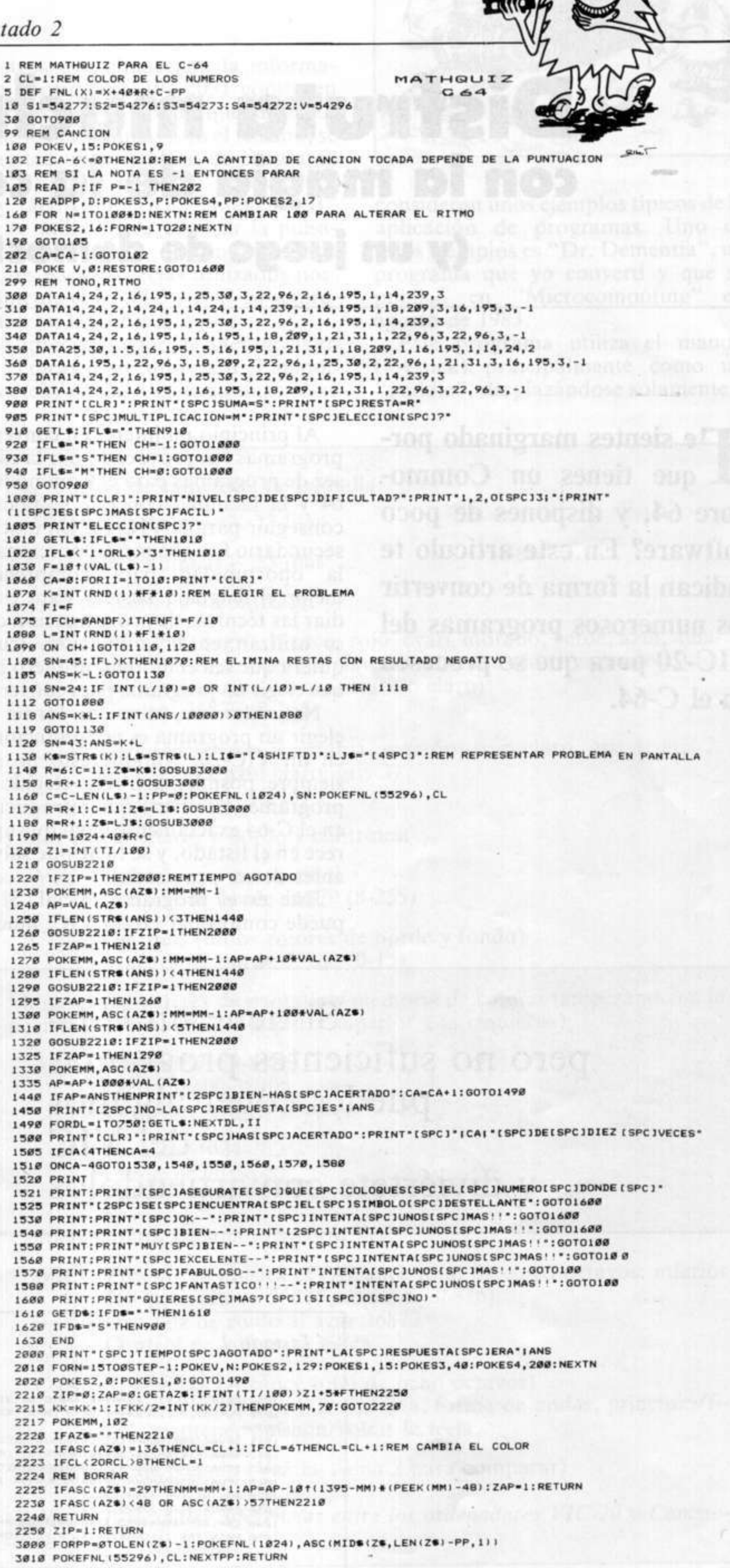
nivel de dificultad (1, 2 ó 3). A continuación presenta el problema en pantalla en posición vertical y sitúa el cursor debajo del dígito de la extrema derecha para que se saque el resultado de dicha columna. Entonces el cursor se va desplazando a la izquierda hasta que se introduzcan todos los dígitos.

Por lo tanto, se daría la respuesta de la suma nueve más cinco, tecleando un cuatro seguido por un uno. El resultado se introduce de derecha a izquierda para simular el proceso utilizado cuando se escribe el problema sobre papel, y es imprescindible hacerlo así cuando se estudian problemas con muchos dígitos cuyas soluciones no han sido memorizadas. Se proporciona una "goma de borrar" para que se pueda cambiar un número mediante el movimiento del cursor. En el ejemplo presentado antes, si se hubiera tecleado un uno primero, se podría borrar mediante la pulsación de la tecla apropiada del cursor, permitiendo la introducción de un nuevo número.

Por supuesto, el tiempo que se tarda en realizar tales correcciones se descuenta del tiempo asignado para cada problema. Si el estudiante empieza a perder el interés, se puede modificar el color del siguiente problema mediante la pulsación de la tecla f7. La canción también puede ser cambiada, preferentemente por el estudiante.

El Listado 1 presenta el programa para el VIC-20, y el Listado 2 para el Commodore 4. Los niños disfrutarán "jugando con el ordenador" mientras refuerzan sus conocimientos matemáticos con este programa.

Listado 2




```
1 REM MATHQUIZ PARA EL C-64
2 CL=1:REM COLOR DE LOS NUMEROS
5 DEF FNL(X)=X+4Ø*R+C-PP
1Ø S1=54277:S2=54276:S3=54273:S4=54272:V=54296
3Ø GOTO9ØØ
99 REM CANCION
1ØØ POKEV,15:POKES1,9
1Ø2 IFCA-6<=ØTHEN21Ø:REM LA CANTIDAD DE CANCION TOCADA DEPENDE DE LA PUNTUACION
1Ø3 REM SI LA NOTA ES -1 ENTONCES PARAR
1Ø5 READ P:IF P=-1 THEN2Ø2
12Ø READPP,D:POKES3,P:POKES4,PP:POKES2,17
16Ø FOR N=1TO1ØØ*D:NEXTN:REM CAMBIAR 1ØØ PARA ALTERAR EL RITMO
17Ø POKES2,16:FORN=1TO2Ø:NEXTN
19Ø GOTO1Ø5
2Ø2 CA=CA-1:GOTO1Ø2
21Ø POKE V,Ø:RESTORE:GOTO16ØØ
299 REM TONO,RITMO
3ØØ DATA14,24,2,16,195,1,25,3Ø,3,22,96,2,16,195,1,14,239,3
31Ø DATA14,24,2,14,24,1,14,24,1,14,239,1,16,195,1,18,2Ø9,3,16,195,3,-1
32Ø DATA14,24,2,16,195,1,25,3Ø,3,22,96,2,16,195,1,14,239,3
34Ø DATA14,24,2,16,195,1,16,195,1,18,2Ø9,1,21,31,1,22,96,3,22,96,3,-1
35Ø DATA25,3Ø,1.5,16,195,.5,16,195,1,21,31,1,18,2Ø9,1,16,195,1,14,24,2
36Ø DATA16,195,1,22,96,3,18,2Ø9,2,22,96,1,25,3Ø,2,22,96,1,21,31,3,16,195,3,-1
37Ø DATA14,24,2,16,195,1,25,3Ø,3,22,96,2,16,195,1,14,239,3
38Ø DATA14,24,2,16,195,1,16,195,1,18,2Ø9,1,21,31,1,22,96,3,22,96,3,-1
9ØØ PRINT"[CLR]":PRINT"[SPC]SUMA=S":PRINT"[SPC]RESTA=R"
9Ø5 PRINT"[SPC]MULTIPLICACION=M":PRINT"[SPC]ELECCION[SPC]?"
91Ø GETL$:IFL$=""THEN91Ø
92Ø IFL$="R"THEN CH=-1:GOTO1ØØØ
93Ø IFL$="S"THEN CH=1:GOTO1ØØØ
94Ø IFL$="M"THEN CH=Ø:GOTO1ØØØ
95Ø GOTO9ØØ
1ØØØ PRINT"[CLR]":PRINT"NIVEL[SPC]DE[SPC]DIFICULTAD?":PRINT"1,2,O[SPC]3;":PRINT"
(1[SPC]ES[SPC]MAS[SPC]FACIL)"
1ØØ5 PRINT"ELECCION[SPC]?"
1Ø1Ø GETL$:IFL$=""THEN1Ø1Ø
1Ø2Ø IFL$<"1"ORL$>"3"THEN1Ø1Ø
1Ø3Ø F=1Ø↑(VAL(L$)-1)
1Ø6Ø CA=Ø:FORII=1TO1Ø:PRINT"[CLR]"
1Ø7Ø K=INT(RND(1)*F*1Ø):REM ELEGIR EL PROBLEMA
1Ø74 F1=F
1Ø75 IFCH=ØANDF>1THENF1=F/1Ø
1Ø8Ø L=INT(RND(1)*F1*1Ø)
1Ø9Ø ON CH+1GOTO111Ø,112Ø
11ØØ SN=45:IFL>KTHEN1Ø7Ø:REM ELIMINA RESTAS CON RESULTADO NEGATIVO
11Ø5 ANS=K-L:GOTO113Ø
111Ø SN=24:IF INT(L/1Ø)=Ø OR INT(L/1Ø)=L/1Ø THEN 1118
1112 GOTO1Ø8Ø
1118 ANS=K*L:IFINT(ANS/1ØØØØ)>ØTHEN1Ø8Ø
1119 GOTO113Ø
112Ø SN=43:ANS=K+L
113Ø K$=STR$(K):L$=STR$(L):LI$="[4SHIFTD]":LJ$="[4SPC]":REM REPRESENTAR PROBLEMA EN PANTALLA
114Ø R=6:C=11:Z$=K$:GOSUB3ØØØ
115Ø R=R+1:Z$=L$:GOSUB3ØØØ
116Ø C=C-LEN(L$)-1:PP=Ø:POKEFNL(1Ø24),SN:POKEFNL(55296),CL
117Ø R=R+1:C=11:Z$=LI$:GOSUB3ØØØ
118Ø R=R+1:Z$=LJ$:GOSUB3ØØØ
119Ø MM=1Ø24+4Ø*R+C
12ØØ Z1=INT(TI/1ØØ)
121Ø GOSUB221Ø
122Ø IFZIP=1THEN2ØØØ:REMTIEMPO AGOTADO
123Ø POKEMM,ASC(AZ$):MM=MM-1
124Ø AP=VAL(AZ$)
125Ø IFLEN(STR$(ANS))<3THEN144Ø
126Ø GOSUB221Ø:IFZIP=1THEN2ØØØ
1265 IFZAP=1THEN121Ø
127Ø POKEMM,ASC(AZ$):MM=MM-1:AP=AP+1Ø*VAL(AZ$)
128Ø IFLEN(STR$(ANS))<4THEN144Ø
129Ø GOSUB221Ø:IFZIP=1THEN2ØØØ
1295 IFZAP=1THEN126Ø
13ØØ POKEMM,ASC(AZ$):MM=MM-1:AP=AP+1ØØ*VAL(AZ$)
131Ø IFLEN(STR$(ANS))<5THEN144Ø
132Ø GOSUB221Ø:IFZIP=1THEN2ØØØ
1325 IFZAP=1THEN129Ø
133Ø POKEMM,ASC(AZ$)
1335 AP=AP+1ØØØ*VAL(AZ$)
144Ø IFAP=ANSTHENPRINT"[2SPC]BIEN-HAS[SPC]ACERTADO":CA=CA+1:GOTO149Ø
145Ø PRINT"[2SPC]NO-LA[SPC]RESPUESTA[SPC]ES";ANS
149Ø FORDL=1TO75Ø:GETL$:NEXTDL,II
15ØØ PRINT"[CLR]":PRINT"[SPC]HAS[SPC]ACERTADO":PRINT"[SPC]";CA;"[SPC]DE[SPC]DIEZ[SPC]VECES"
15Ø5 IFCA<4THENCA=4
151Ø ONCA-4GOTO153Ø,154Ø,155Ø,156Ø,157Ø,158Ø
152Ø PRINT
1521 PRINT:PRINT"[SPC]ASEGURATE[SPC]QUE[SPC]COLOQUES[SPC]EL[SPC]NUMERO[SPC]DONDE[SPC]"
1525 PRINT"[2SPC]SE[SPC]ENCUENTRA[SPC]EL[SPC]SIMBOLO[SPC]DESTELLANTE":GOTO16ØØ
153Ø PRINT:PRINT"[SPC]OK--":PRINT"[SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO16ØØ
154Ø PRINT:PRINT"[SPC]BIEN--":PRINT"[2SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO16ØØ
155Ø PRINT:PRINT"MUY[SPC]BIEN--":PRINT"[SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO1ØØ
156Ø PRINT:PRINT"[SPC]EXCELENTE--":PRINT"[SPC]INTENTA[SPC]UNOS[SPC]MAS!!":GOTO1Ø Ø
157Ø PRINT:PRINT"[SPC]FABULOSO--":PRINT"INTENTA[SPC]UNOS[SPC]MAS!!":GOTO1ØØ
158Ø PRINT:PRINT"[SPC]FANTASTICO!!!--":PRINT"INTENTA[SPC]UNOS[SPC]MAS!!":GOTO1ØØ
16ØØ PRINT:PRINT"QUIERES[SPC]MAS?[SPC](SI[SPC]O[SPC]NO)"
161Ø GETD$:IFD$=""THEN161Ø
162Ø IFD$="S"THEN9ØØ
163Ø END
2ØØØ PRINT"[SPC]TIEMPO[SPC]AGOTADO":PRINT"LA[SPC]RESPUESTA[SPC]ERA";ANS
2Ø1Ø FORN=15TOØSTEP-1:POKEV,N:POKES2,129:POKES1,15:POKES3,4Ø:POKES4,2ØØ:NEXTN
2Ø2Ø POKES2,Ø:POKES1,Ø:GOTO149Ø
221Ø ZIP=Ø:ZAP=Ø:GETAZ$:IFINT(TI/1ØØ)>Z1+5*FTHEN225Ø
2215 KK=KK+1:IFKK/2=INT(KK/2)THENPOKEMM,7Ø:GOTO222Ø
2217 POKEMM,1Ø2
222Ø IFAZ$=""THEN221Ø
2222 IFASC(AZ$)=136THENCL=CL+1:IFCL=6THENCL=CL+1:REM CAMBIA EL COLOR
2223 IFCL<2ORCL>8THENCL=1
2224 REM BORRAR
2225 IFASC(AZ$)=29THENMM=MM+1:AP=AP-1Ø↑(1395-MM)*(PEEK(MM)-48):ZAP=1:RETURN
223Ø IFASC(AZ$)<48 OR ASC(AZ$)>57THEN221Ø
224Ø RETURN
225Ø ZIP=1:RETURN
3ØØØ FORPP=ØTOLEN(Z$)-1:POKEFNL(1Ø24),ASC(MID$(Z$,LEN(Z$)-PP,1))
3Ø1Ø POKEFNL(55296),CL:NEXTPP:RETURN
```

Disfruta mucho más con la magia de la conversión

(y un juego de demostración)

Te sientes marginado porque tienes un Commodore 64, y dispones de poco software? En este artículo te indican la forma de convertir los numerosos programas del VIC-20 para que se procesen en el C-64.

Al principio mi interés en convertir programas fue provocado por la escasez de programas para el Commodore 64 y la facilidad con que se podían conseguir para el VIC-20. Un motivo secundario fue que me proporcionaba la oportunidad para comprender mejor el lenguaje Basic y para estudiar las técnicas de programación que se utilizan en la animación. Cualquiera que sea el motivo, primero hay que elegir un programa para convertir.

Naturalmente, la mejor forma de elegir un programa es verlo funcionar en un VIC-20. Aunque esto no sea siempre posible, una vez elegido el programa, éste tiene que introducirse en el C-64 exactamente igual que aparece en el listado, y se tiene que salvar antes de ser procesado.

Este es el programa inicial. Si se puede comprobar en un VIC, mucho mejor y si se detecta algún fallo resulta útil apuntar el número de la sentencia. Es probable que los mismos fallos salgan en el C-64.

> Si tiene un C-64
> pero no suficientes programas
> puedes coger algunos
> del VIC
> y diviértete convirtiéndolos

Las Principales Diferencias de Programa

El VIC-20 y el C-64 se parecen en muchos aspectos, pero existen diferencias en los métodos que emplean para controlar los registros utilizados para producir la acción en la pantalla y en el sonido. Los procedimientos de lectura del teclado y del joystick también son diferentes. (Las Tablas 1 y 2 presentan un resumen).

Las sentencias "Print" o "Poke" controlan los caracteres de la pantalla y los colores de los caracteres. Las sentencias "Print" en un VIC se pueden imprimir en la pantalla de un C-64, realizando unas modificaciones mínimas dada la diferencia del tamaño de la pantalla. Normalmente, un cambio es necesario si una sentencia "Print" termina en punto y coma, ya que se escribe de forma diferente en la pantalla más ancha del C-64.

Las sentencias "Poke" siempre se tienen que modificar para que se ajuste la posición de memoria del VIC a la posición equivalente en el C-64. Yo he inventado un programa sencillo (El Listado 1) que convierte la posi-

Listado 1.

```
1000 PRINT"[CLR]":REM****CAMBIO DE DIRECCIONES DE PANTALLA DEL VIC 20 AL C-64
1010 POKE53281,1:FLAG=0:VA=0:INPUT"*[SPC]DIRECCION[SPC]VIC[SPC]20";VA:IFVA=0THE
NSTOP
1011 IFVA<7680THENPRINT"[RVSON]DIRECCION[SPC]DEMASIADO[SPC]BAJA":GOTO1010
1012 IFVA>8185ANDVA<38400THENPRINT"[RVSON]DIRECCION[SPC]NO[SPC]VALIDA":GOTO1010
1013 IFVA>38905THENPRINT"[RVSON]DIRECCION[SPC]DEMASIADO[SPC]ALTA":GOTO1010
1015 IFVA>38399ANDVA<38906THENVA=VA-30720:FLAG=1
1020 RO=INT((VA-7680)/22)
1030 CO=VA-7680-22*RO
1040 CA=1024+RO*40+CO:IFFLAG=1THENCA=CA+54272
1050 PRINTTAB(21)CHR$(145);"DIRECCION[SPC]C-64";CA:GOTO1010
```



ción de caracteres en el VIC en la posición para el C-64. Se puede determinar la posición del carácter "Poke" en color añadiendo un "offset" a la posición del carácter. El "offset" para el VIC es 30720 y el "offset" para el C-64 es 54272.

En el VIC un sólo registro controla los colores del borde y el fondo de la pantalla, mientras en el C-64 se emplean dos registros para controlar los colores. El formato de la sucesión de los colores de los "Poke" utilizado por el VIC es el más complicado. El valor más bajo de "Poke" es 8 y representa un fondo negro con un borde negro.

A medida que este valor se incrementa por uno, el color del borde se incrementa por uno en el orden de la clave de color en el teclado (de negro a amarillo). Para aumentar el color del fondo por uno (en el orden presentado en la Tabla 1), se necesita un incremento de 16. Por ejemplo, 24 produce un borde negro con un fondo blanco; el "Poke" más alto, 255, produce un fondo de amarillo claro y un borde amarillo.

Dado que los códigos de color "Poke" en el C-64 son constantes (0-15), el efecto producido se ve en seguida que se hace un "Poke" cada número de color en los dos registros. Cuando se hace un "Poke" cero en el registro del borde en el C-64, aparece un borde negro.

El VIC-20 y el C-64 también producen el sonido de dos maneras diferentes. El VIC tiene cuatro registros dedicados al sonido. Tres de las voces son musicales, cada una con un determinado alcance de frecuencias, mientras que la cuarta produce ruido al azar. Las voces del VIC pueden estar en "on" o en "off", y se controlan especialmente por las posiciones del volumen y de la frecuencia (128-255).

El C-64 posee un sintetizador de sonido de tres voces. Cada voz es capaz de producir ruidos y música. El sonido exacto que se produce se controla mediante las posiciones de volumen, frecuencia, forma de ondas, principio/final y mantener la tecla pulsada/soltar la tecla. El grado de control alcanzable sobrepasa el que se necesita para la mayoría de los juegos o programas.

Diferencias de Entrada

Físicamente los teclados de los dos ordenadores son idénticos, pero al pulsar una tecla determinada, los registros que almacenan la información (posiciones 197-203) contienen distintos valores. Dado que la mayoría de los programas leen el teclado, se necesitan unos cambios de programa para producir el efecto deseado. Es muy fácil cambiar la sentencia "If" que se utiliza para detectar la pulsación de una tecla determinada. Los valores para las teclas utilizados normalmente en el VIC y el C-64 se presentan en la Tabla 2.

Se aprecian mejor las diferencias en las técnicas de lectura para el mando (joystick) en las dos máquinas si se consideran unos ejemplos típicos de la aplicación de programas. Uno de estos ejemplos es "Dr. Dementia", un programa que yo convertí y que se publicó en "Microcomputing" en agosto de 1983.

Este programa utiliza el mando (joystick) principalmente como un "paddle", desplazándose solamente a

Disposición de caracteres de pantalla:

VIC 22 columnas × 23 filas
C-24 40 columnas × 25 filas

Fondo de pantalla y controles:

VIC Dieciséis (negro, blanco, rojo, cyan, morado, verde, azul, amarillo, naranja, naranja claro, rosa, cyan claro, morado claro, verde claro, azul claro, amarillo claro)
Control 36879 (8-255)

C-24 Dieciséis (negro a naranja, marrón, rojo claro, gris 1, gris 2, verde claro, azul claro, gris 3)
Control 53281 (0-15)

Color de caracteres y borde y controles:

VIC Ocho (negro a amarillo)
Control de borde 36879 (8-255)

C-64 Dieciséis (todos colores de borde y fondo)
Control de borde 53280 (0-15)

Mapas de caracteres de pantalla y memoria de colores (empezando en la posición de pantalla en la parte superior a la izquierda):

VIC Carácter 7680-8185
Color 38400-38905
Carácter "offset" a color 30720

C-64 Carácter 1024-2023
Color 55296-56295
Carácter "offset" a color 54272

Chip generador de sonido y controles:

VIC Tres voces musicales (Alcance total de cinco octavos: inferior 36874, mediano 36875, superior 36876)
Una voz de ruido al azar 36877
Control de volumen 36878

C-64 Tres voces (Alcance total de ocho octavos)
Controles: volumen, frecuencia, forma de ondas, principio/final, mantener pulsada/soltar la tecla.

Disposición de teclado (Vea la Tabla 2 para comparar)

Tabla 1. Principales diferencias entre los ordenadores VIC-20 y Commodore 64.

la izquierda y a la derecha. Se puede utilizar una programación semejante para obtener movimiento en cualquiera de las ocho direcciones. Las sentencias de programa original del VIC y el programa convertido del C-64 se presentan a continuación:

```
VIC   POKE 37154,127; PK=PEEK(37152)
      AND 128: JO=—(PK=0): POKE
      37154,255: PK=PEEK(37151): J2=
      —((PK AND 16)=0): fb= —((PK
      AND 32)=0)
C-64  JV=PEEK(56320) AND 15: FB=
      —((PEEK(56320)AND 16)=0)
```

El primer "Poke" del VIC fija uno de los dos registros de la dirección de datos en la modalidad de entrada. Esto permite la lectura del interruptor 3 (derecha) del mando (joystick) y se analiza una función lógica "AND" para establecer un valor para PX. Si el mando (joystick) se desplaza a la derecha, J0 se fijaría en uno mediante la sentencia lógica AND. Si el joystick estuviera estacionario, J0 sería cero.

Dado que el registro de la dirección de datos tiene dos funciones en el VIC, se tiene que restaurar o hacerle un POKE a 255 después de completar la lectura del interruptor 3. Los otros tres interruptores del mando (joystick) se leen haciendo un PEEK en la posición 37151. Se determina el estado de los interruptores realizando la función lógica "AND" en el valor leído por las potencias de 2. Si se realiza la función lógica "AND" en PEEK con 4 se detecta el interruptor que se desplaza hacia arriba. De modo semejante, si se realiza la función lógica AND con el 8 y 16, se detectan las direcciones hacia abajo y a la izquierda, respectivamente.

El movimiento en diagonal se detecta cuando dos interruptores se cierran a la vez. Se detecta el movimiento del botón de disparo mediante la realización de la función lógica AND con 128. Las sentencias lógicas para J2 y FB son ejemplos de la detección de la dirección a la izquierda y el botón de disparo. El VIC dispone de otros registros para leer el mando, pero los que se presentan aquí son los que se utilizan normalmente. Los registros alternativos son 37137 y 37139.

El joystick del C-64 se lee simplemente haciendo un "PEEK" en una de las dos posiciones (56320 para la Puerta 2; 56321 para la Puerta 1). El valor donde se ha hecho el PEEK da 14, 13, 11 y 7 para el movimiento del joystick hacia arriba, hacia abajo a la derecha y a la izquierda, respectivamente cuando se realiza la función lógica "AND" con 15. El movimiento del mando (joystick) en diagonal produce valores de 5, 6, 9 y 10. El botón de disparo se detecta realizando la función lógica "AND" del PEEK con 16.

El procedimiento general para modificar una porción del programa del mando en el VIC para que funcione en el C-64 es la que se presenta a continuación:

—Explorar el listado para los registros mencionados y el valor que utilizan para "AND".

—Fijar la lógica del programa que utiliza la entrada del mando (joystick).

—Modificar las sentencias VIC a las simples sentencias equivalentes en el C-64.

Una demostración de Prueba

La mejor forma de saber la manera en que todas estas diferencias afectan un programa típico es realizando un ejercicio de conversión. Para este ejercicio, he elegido un juego de "Public Domain", "Artillery" (Artillería), escogido porque las modificaciones de conversión son típicas de la mayoría de los programas sencillos de Basic. Los Listados 2 y 3 presentan el programa original de VIC y el programa convertido del C-64. El derecho para utilizar este juego fue otorgado por Bill Munch de Public Domain, Inc., West Milton, Ohio.

El método utilizado proporciona un programa del C-64 que funciona de modo semejante que el programa del VIC. Sin embargo, el uso del procedimiento detallado no produce en el C-64 SID (Sound Interface Device) (Dispositivo de Sonido de la Interface) exactamente los mismos sonidos que los que se producen en el VIC. Si se desea mayor fidelidad, los registros SID tienen que ser ajustados, que a su vez significa ajustar los controles de

TECLAS STANDARD

FILA 1			FILA 2			FILA 3			FILA 4		
Tecla	VIC	C-64	Tecla	VIC	C-64	Tecla	VIC	C-64			
—	8	57	Q	48	62	A	17	10	Z	33	12
1	0	56	W	9	9	S	41	13	X	26	23
2	56	59	E	48	14	D	18	18	C	34	20
3	1	8	R	10	17	F	42	21	V	27	31
4	57	11	T	50	22	G	19	26	B	35	28
5	2	16	Y	11	25	H	43	29	N	28	39
6	58	19	U	51	30	J	20	34	M	36	36
7	3	24	I	12	33	K	44	37	,	29	47
8	59	27	O	52	38	L	21	42	.	37	44
9	4	22	P	13	41	:	45	45	/	30	55
0	30	35		53	46	;	22	50			
+	5	40	*	14	49	=	46	53			
—	61	43	^	54	54						
£	6	48									

Teclas de Función y Control Especial

Tecla	VIC	C-64	Tecla	VIC	C-64
Clear—Home	62	51	f1	39	4
Insert—Delete	62	0	f3	47	5
Return	15	1	f5	55	6
Space Bar	32	6	f7	63	3
Cursor—Vertical	31	7	Cursor—Horizontal	23	2

Nota: Cuando no hay teclas pulsadas las posiciones de memoria 197 y 203 contienen 64.

Tabla 2. Diferencias del teclado del VIC-20 y el C-64.



frecuencia, forma de ondas, principio/final y mantener pulsada/soltar tecla.

Lo difícil de la conversión es seguir el flujo del programa. Como los programadores del VIC disponen de una memoria de máquina muy limitada utilizan pocas sentencias "Remark". Una manera de rastreo es colocando un "Stop" antes y después de cada sentencia de bifurcación (If, Go Sub, Go To, etc.) en el programa inicial. Luego se modifican las sentencias antes del "Stop" y se retiran a medida que cada división pasa sin problemas obteniendo el proceso del programa.

En "Artillería" dos jugadores disparan cañones por encima de una montaña que cambia de forma al azar. Para que el juego resulte más complicado hay que tener en cuenta el factor del viento al elegir el ángulo de disparo del cañón. También hay que ajustar la carga, o la pólvora.

Las líneas o sentencias del listado del C-64 que se modifican serán tratadas por grupo a medida que surjan cambios comunes de sonido, pantalla, teclado, etc.

Líneas 0 y 75: La función DEF en la línea 0 se utiliza a lo largo del programa para hacer un "Poke" de varios símbolos gráficos en las posiciones de la memoria de pantalla, según los valores actuales de X e Y. Esta función DEF también podría haber sido utilizada por el programador del VIC en la línea 75, pero no se hizo así. Las diferencias de la ecuación en ambas líneas se relacionan al comienzo de la memoria de caracteres de la pantalla (VIC, 7680; C-64, 1024) y al número de filas y columnas (VIC, 22 columnas × 23 filas; C-64, 40 columnas × 25 filas).

Línea 4: Esta subrutina inicial (líneas 3500—3640) trata, en principio, con los registros de teclas y los "array". La primera línea establece los límites de memoria de los registros SID e inicializa tres variables (BO, BA, CU) para controlar los colores de pantalla para el borde, el fondo y el cursor. La siguiente línea borra el chip generador de sonido, SID, según las recomendaciones de Commodore.

El programa inicia el establecimiento de variables que se utilizan para seguir la pista de los registros de teclas SID; a continuación inicializa las dos voces SID utilizadas en el programa. Las voces 1 y 2 imitan las voces de música y ruidos del VIC. La voz 3 se inicializa pero no se utiliza en el programa.

Para la mayoría de los programas se pueden utilizar valores fijos para principio/final (AD), mantener pulsada/soltar la tecla (SR), y de baja frecuencia (LF). Yo utilizo el comando "Poke" estandard, y lo modifico solamente si la calidad del sonido es pobre. Finalmente, la subrutina fija el ángulo inicial y los valores de pólvora para cada jugador.

Línea 5: Los comandos "Poke" del C-64 desempeñan las mismas funciones que los "Poke" del VIC; fijan los colores deseados para el borde y el fondo de la pantalla y para el cursor. Como hemos mencionado antes, en el VIC el valor almacenado en la posición 36879 controla los colores del borde y el fondo.

En el VIC, los comandos "Poke" producen los colores blanco, blanco y azul para el borde, fondo y cursor. Los colores del C-64 son blanco, blanco y gris 1.

En la versión del C-64, KX se fija en 65272. Esto es el "offset" entre las posiciones de los caracteres y la memoria de la pantalla en color. Se utilizará más adelante en el programa y se necesita para mantener la estructura del programa.

A veces, es necesario abreviar las palabras clave y añadir variables o sentencias para reducir las sentencias del VIC (longitud máxima de 88 caracteres) para que quepan en el C-64 (longitud máxima de 80 caracteres).

Línea 7: Esta subrutina mejorada (las líneas 4.000-4.100) escribe las instrucciones de juego en la pantalla. Retorna a la sentencia 7 cuando el "Peek" o el valor almacenado en la posición 197 (al pulsar una tecla) se fija en 60. Esto ocurre cuando se pulsa la barra de espaciado.

Línea 11: Se realizan mejoras en la versión del C-64 para fijar los valores actuales de ángulo (A) y pólvora (V) a los valores antes seleccionados por el jugador 1 ó 2. Los parámetros actua-

VIC Address	Función	C-64 Address
197 ó 203	Tecla pulsada	197 ó 203
646	Color del cursor	646
7680—8185	Posición de memoria de caracteres de pantalla	1024—2023
36874-36878	Registro de sonido	54272—54296
36879	Registro de color de pantalla (fondo y borde)	53280 y 53281
37137	Registro de salida alternativa del mando (joystick)	—
37139	Registro de dirección de datos alternativa del mando (joystick)	—
37151	Registro de salida principal del mando (joystick) para interruptores 0, 1 y 2	56320 y 56321 (all switches)
37152	Registro de salida principal del mando (joystick) para interruptor 3 (derecha)	—
37154	Registro de dirección de datos principal del mando (joystick)	—
38400—38905	Posiciones de memoria de color de caracteres de pantalla	55296—56295

Tabla 3. Direcciones importantes del VIC-20 y las equivalencias en el C-64.

les se salvan en la sentencia 27 y se inicializan en la sentencia 3630.

Línea 13: Se necesita el "Poke" adicional en el programa del C-64 para que el cursor vuelva al color de gris 1 después de que las montañas y los cañones se imprimen en la subrutina 3200.

Líneas 14, 15, 18, 19 y 24: Estas líneas se tienen que modificar dadas las diferencias en los teclados. Un "Peek" de la posición 197 ó 203 retorna valores diferentes en los dos ordenadores cuando se pulsa la misma tecla. La Tabla 2 compara estos valores.

Línea 24: En la versión del C-64, se elimina el "Poke" de sonido equivalente al del VIC. El Poke del VIC sube el volumen al máximo, 15. Si este "Poke" no se eliminara, el SID emitiría un ruido seco y repetitivo. Se retrasa la conversión del ángulo de disparo elegido en radianes hasta que se salva como AS(PL) en la sentencia 27.

Líneas 25, 26, 65, 67, 81, 120, 2001, 2003, 2060, 2070, 2076 y 2078: Los "Poke" de sonido del VIC en estas sentencias se modificaron para reproducir la música o sonidos equivalentes. Para que el procedimiento de conversión resultara fácil, decidí fijar la mayoría de los registros SID en la subrutina 3500; sin embargo, se tienen que controlar un mínimo de tres. Esto es un registro más que los que se necesitan en el VIC.

Los registros para el C-64 son VL (volumen), HF (alta frecuencia) y WF (forma de ondas). Los registros del VIC son volumen (36878) y frecuencia, o voz, (36874-36877). Los registros de volumen de ambos ordenadores se manejan de forma idéntica, pero los registros de frecuencia tienen diferentes alcances de "Poke". Se puede hacer un "Poke" en los registros del VIC con valores de entre 128 y 255, mientras que el alcance del C-64, de 0 a 255, es el doble de grande.

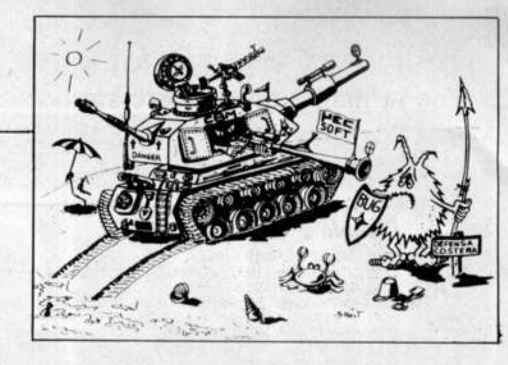
Los registros de forma de ondas del C-64 se utilizan para seleccionar el tipo de sonido producido y para apagarlo y encenderlo. Yo fijé la voz 1 para ruido y la voz 2 para música. La forma de onda para la voz 1, WF(1), contiene o 129 (ruido encendido) o 128 (ruido apagado). De modo semejante, el registro de forma de ondas de la voz 2 contiene o 17 (música encendida) o 16 (música apagada).

Los registros de frecuencia de VIC controlan el tipo de sonido producido. Los tres primeros registros (el registro más bajo es la voz del octavo

El Listado 2. Programa original "Artillería" para el VIC-20. Este listado no funciona en un VIC-20 no ampliado a no ser que se quiten las sentencias REM.

Artillería VIC 20.

```
0 CLR:DEFFNCL(R)=7680.5+INT(22.5-Y)*22+X:X=RND(-TI):PL=1
1 REM*ESTO ES ARTILLERIA 20 PROGRAMA ORIGINAL DE 'PUBLIC DOMAIN'* VER COMENTARIO
S
5 PRINT"[CLR]":POKE36879,25
9 GOSUB3000:W=(2*RND(1)):W=W/2:R=INT(2*RND(1)+1):IFR=1THENW=-(W)
10 G=9.8:H=.04:T=0:VX=0:VY=0:X=0:Y=0
11 R=INT(2*RND(1)+1):LP=0:A=45:V=20:IFPL=2THENLP=11
12 RR=(RND(1)):R=INT(2*RND(1)+1):IFRR>.2THENRR=.2
13 GOSUB3200
14 IFPEEK(197)=39THENA=A+1
15 IFPEEK(197)=47THENA=A-1
16 IFA<0THENA=0
17 IFA>90THENA=90
18 IFPEEK(197)=55THENV=V+1
19 IFPEEK(197)=63THENV=V-1
20 IFV<1THENV=1
21 IFV>40THENV=40
22 PRINT"[HOM]"SPC(LP+8)"[2SPC]":PRINT"[HOM]"SPC(LP)"ANGULO[SPC]=[3SPC][3CRSRL]"
A
23 PRINT"[HOM][CRSRD]"SPC(LP+8)"[2SPC]":PRINT"[HOM][CRSRD]"SPC(LP)"POLVORA=[3SPC
][3CRSRL]"V
24 IFPEEK(197)=15THENA=π/180*A:POKE36878,15:GOTO37
25 POKE36878,WH:POKE36877,WD:GOTO14
37 POKE36877,0:PRINT"[HOM][44SPC]"
38 PRINT"[HOM][2CRSRD][22SPC]"
39 ONPLGOTO40,50
40 X=X1:Y=Y1+1:XI=X:YI=Y:PC=FNCL(0):S=1:GOTO60
50 S=-1:X=X2:Y=Y2+1:XI=X:YI=Y:PC=FNCL(0)
60 VY=SIN(A)*V:VX=S*(COS(A)*V):FORI=228TO168STEP-1:POKE36877,I:NEXT:POKE36877,0
70 T=T+H:X=XI+VX*T-W*T↑2:Y=YI+VY*T-G*T↑2
75 PK=7680.5+INT(22.5-Y)*22+X:PK=INT(PK):POKEPC,32
80 IFY>22THEN115
81 IFX<0ORX>21THENPOKE36876,0:GOTO2004
85 PC=PK:IFY<0THEN2000
90 IFPEEK(PK)<>32THEN1000
110 POKEPK+30720,0:POKEPK,46
115 IFY*2+137=>255ORY*2+137<0THEN130
120 POKE36876,Y*2+137
130 IFY>0THEN70
140 GOTO2000
1000 IFPEEK(PK)=233ORPEEK(PK-22)=233THENPOKEPK,42:H1=H1+1:H=1:PL=1:GOTO2060
1110 IFPEEK(PK)=223ORPEEK(PK-22)=223THENPOKEPK,42:H2=H2+1:H=1:PL=2:GOTO2060
2000 KK=PEEK(PK):POKEPK,32:I=SIN(I):POKEPK,KK:POKEPK,32:I=SIN(I):POKEPK,KK
2001 POKE36876,0:FORI=1TO50:POKE36877,128+I:NEXT
2002 POKEPK,32:I=SIN(I):POKEPK,KK:POKEPK,32:I=SIN(I):POKEPK,KK
2003 FORI=1TO50:POKE36877,178-I:NEXT:POKE36877,0
2004 IFPL=1THENPL=2:GOTO2010
2005 IFPL=2THENPL=1:GOTO2010
2010 IFH=1THENH=0:PRINT"[CLR]":GOTO9
2020 GOTO10
2060 K=30720:POKE36876,0:G=7:S0=36877:FORI=1TO10
2070 POKEPK+K,G:POKES0,128:S=SIN(I):POKEPK-22+K,G
2072 POKEPK-43+K,G:POKES0,158:POKEPK-45+K,G
2076 POKES0,188:S=SIN(I):S=SIN(I):IFG=42THENG=32:GOTO2078
2077 G=42:K=0
2078 NEXTI:POKES0,128:FORI=1TO100:NEXT:POKES0,0:GOTO2004
2087 G=42:K=0
3000 PRINT"[HOM][2CRSRD][RVSON][RED]SC="H1"[HOM][2CRSRD][11CRSRR][PUR]SC="H2
3003 R=INT(6*RND(1)+1):FORY=0TOR-1:FORX=0TO5:POKEFNCL(0)+30720,5
3005 POKEFNCL(0),224:NEXTX:NEXTY
3007 Y1=Y:X1=INT(4*RND(1)+1):X=X1:POKEFNCL(0)+30720,2:POKEFNCL(0),223
3010 FORY=0TO5:FORX=6TO15:POKEFNCL(0)+30720,5
3020 POKEFNCL(0),160:NEXTX:NEXTY:R=INT(6*RND(1)+1):FORY=0TOR-1:FORX=16TO21
3030 POKEFNCL(0)+30720,5:POKEFNCL(0),224:NEXTX:NEXTY
3035 Y2=Y:X2=INT(3*RND(1)+17):X=X2:POKEFNCL(0)+30720,4:POKEFNCL(0),233
3040 P1=INT(3*RND(1)+1):P1=P1+5:P2=INT(3*RND(1)+1):P2=16-P2:I=INT(2*RND(1)+1)
3050 IFI=1THENL1=255:GOTO3060
3055 L1=254
3060 I=INT(2*RND(1)+1):IFI=1THENL2=97:GOTO3070
3065 L2=252
3070 X=P1:Y=6:POKEFNCL(0)+30720,5:POKEFNCL(0),L1
3071 FORX=P1+1TOP2-1:Y=6:POKEFNCL(0)+30720,5
3075 POKEFNCL(0),160:NEXTX:X=P2:POKEFNCL(0)+30720,5:POKEFNCL(0),L2
3080 FORY=7TO17:RR=INT(6*RND(1)+1)
3085 IFRR=1THENA1=P1+2:L1=254
3090 IFRR=2THENA1=P1+1:L1=254
3095 IFRR>2THENA1=P1:L1=225
3100 RR=INT(6*RND(1)+1):IFRR=1THENA2=P2-2:L2=252
3110 IFRR=2THENA2=P2-1:L2=252
3120 IFRR>2THENA2=P2:L2=97
3130 IFA2<A1ORA1=A2THEN3160
3140 X=A1:POKEFNCL(0)+30720,5:POKEFNCL(0),L1:FORX=A1+1TOA2-1:POKEFNCL(0)+30720,5
3145 POKEFNCL(0),160:NEXTX:X=A2:POKEFNCL(0)+30720,5:POKEFNCL(0),L2
3150 P1=A1:P2=A2:IFTT=1THEN3160
3155 NEXTY
3160 PX=P1:PY=Y:RETURN
3200 IFR=1THENRR=-(RR)
3210 W=W+(RR/INT(6*RND(1)+4))
3220 IFW<-2THENW=-2
3225 IFW>2THENW=2
3235 WD=240:WH=INT(ABS(W*8)):IFWH>15THENWH=15
3240 IFW>0THEN3270
3260 X=PX:Y=PY:POKEFNCL(0)+30720,6:POKEFNCL(0),103:Y=Y+1
3261 POKEFNCL(0)+30720,6:POKEFNCL(0),103
3262 POKEFNCL(0)+30721,6:POKEFNCL(0)+1,102:GOTO3300
3270 X=PX:Y=PY:POKEFNCL(0)+30720,6:POKEFNCL(0),103:Y=Y+1
3271 POKEFNCL(0)+30720,6:POKEFNCL(0),102
3272 POKEFNCL(0)+30721,6:POKEFNCL(0)+1,32
3300 PRINT"[HOM][22CRSRD]"SPC(5)"[RVSON][GRN][12SPC]";
3301 PRINT"[HOM][2CRSRD][RVSON][RED]SC="H1"[HOM][2CRSRD][11CRSRR][PUR]SC="H2
3302 PRINT"[HOM][22CRSRD]"SPC(5)"[RVSON][GRN]VIENTO"INT(ABS(W*80))"KMH[BLK]";:RE
TURN
```



Listado 3. Programa de "Artillería" convertido para C-64.

Artillería C-64.

```
0 CLR DEFFNCL(R)=1024.5+INT(24.5-Y)*40+X X=RND(-TI) PL=1
1 REM*ESTO ES ARTILLERIA 64 PROGRAMA CONVERTIDO DE 'PUBLIC DOMAIN'
4 GOSUB3500 REM*RUTINA DE INICIALIZACION
5 PRINT"[CLR]" POKEB0,1 POKEBA,1 POKECU,11 KX=54272
7 GOSUB4000 REM*INSTRUCCIONES
9 GOSUB3000 W=(2*RND(1)) W=W/2 R=INT(2*RND(1)+1) IFR=1THENW=-(W) REM*PREFIJAR MT
10 G=9.8 H=.04 T=0 VX=0 VY=0 X=0 Y=0
11 R=INT(2*RND(1)+1) LP=0 A=AS(1) V=VS(1) IFPL=2THENLP=11 A=AS(2) V=VS(2)
12 RR=(RND(1)) R=INT(2*RND(1)+1) IFRR>.2THENRR=.2
13 GOSUB3200 POKECU,11 REM*OBTENER VIENTO Y VISUALIZAR
14 IFPEEK(197)=4THENA=A+1
15 IFPEEK(197)=5THENA=A-1
16 IFA<0THENA=0
17 IFA>90THENA=90
18 IFPEEK(197)=6THENV=V+1
19 IFPEEK(197)=3THENV=V-1
20 IFV<1THENV=1
21 IFV>40THENV=40
22 PRINT"[HOM]"SPC(LP+8)"[2SPC]" PRINT"[HOM]"SPC(LP)"ANGULO[SPC]=[3SPC][3CRSRL]"
A
23 PRINT"[HOM][CRSRD]"SPC(LP+8)"[2SPC]" PRINT"[HOM][CRSRD]"SPC(LP)"POLVORA=[3SPC
][3CRSRL]"V
24 IFPEEK(197)=1GOTO27
25 POKEVL,WH POKEWF(1),129 POKEHF(1),(WD-127)*2 GOTO14 REM*SONIDO VIENTO 'ON'
27 AS(PL)=A VS(PL)=V A=π/180*A REM*SELECCIONES REPRESENTACION
28 PRINT"[HOM][3CRSRD]"SPC(27)"JUGADOR[SPC]#[SPC]1"
29 PRINT"[HOM][5CRSRD]"SPC(27)"ANGULO[SPC]="AS(1)
30 PRINT"[HOM][6CRSRD]"SPC(27)"POLVORA="VS(1)
31 PRINT"[HOM][9CRSRD]"SPC(27)"JUGADOR[SPC]#[SPC]2"
32 PRINT"[HOM][11CRSRD]"SPC(27)"ANGULO[SPC]="AS(2)
33 PRINT"[HOM][12CRSRD]"SPC(27)"POLVORA="VS(2) WP=INT(W*80) IFWP<0THENWP=WP+1
34 PRINT"[HOM][15CRSRD]"SPC(27)"VIENTO[SPC]=[9SPC]"
35 PRINT"[HOM][15CRSRD]"SPC(27)"VIENTO[SPC]="WP
36 POKEWF(1),128 REM*SONIDO DE VIENTO 'OFF'
37 PRINT"[HOM][68SPC]"
38 PRINT"[HOM][2CRSRD][66SPC]"
39 ONPLGOTO40,50
40 X=X1 Y=Y1+1 XI=X YI=Y PC=FNCL(0) S=1 GOTO60
50 S=-1 X=X2 Y=Y2+1 XI=X YI=Y PC=FNCL(0)
60 VY=SIN(A)*V VX=S*(COS(A)*V) REM*SONIDO DE BALA 'ON'
65 POKEVL,15 POKEWF(1),129 FORI=(228-127)*2TO(168-127)*2STEP-2 POKEHF(1),I NEXT
67 POKEWF(1),128 REM*SONIDO DE BALA 'OFF'
70 T=T+H X=XI+VX*T-W*T↑2 Y=YI+VY*T-G*T↑2 REM*RUTINA MOVIMIENTO PROYECTIL70-130
75 PK=1024.5+INT(24.5-Y)*40+X PK=INT(PK) POKEPC,32
80 IFY>22THEN115 REM*PROYECT'L SOBRE PANTALLA SOLO SONIDO
81 IFX<0ORX>21THENPOKEWF(2),16 GOTO2004 REM*FUERA DE LIMITE 'X' SONIDO 'OFF'
85 PC=PK IFY<0THEN2000
90 IFPEEK(PK)<>32THEN1000
110 POKEPK+KX,0 POKEPK,46
115 IF(Y*2+137-127)*2>255OR(Y*2+137-127)*2<0THEN130 REM*VERIFICA POKE SONIDO
120 POKEHF(2),(Y*2+137-127)*2 POKEWF(2),17 REMSONIDO PROYECTIL 'ON'
130 IFY>0THEN70
140 GOTO2000
1000 IFPEEK(PK)=233ORPEEK(PK-40)=233THENPOKEPK,42 H1=H1+1 H=1 PL=1 GOTO2060
1110 IFPEEK(PK)=223ORPEEK(PK-40)=223THENPOKEPK,42 H2=H2+1 H=1 PL=2 GOTO2060
2000 KK=PEEK(PK) POKEPK,32 I=SIN(I) POKEPK,KK POKEPK,32 I=SIN(I) POKEPK,KK
2001 POKEWF(2),16 POKEWF(1),129 FORI=1TO50 POKEHF(1),I*2 NEXT REM*BANDERA TOCADA
2002 POKEPK,32 I=SIN(I) POKEPK,KK POKEPK,32 I=SIN(I) POKEPK,KK
2003 FORI=1TO50 POKEHF(1),(178-127-I)*2 NEXT POKEWF(1),128
2004 IFPL=1THENPL=2 GOTO2010
2005 IFPL=2THENPL=1 GOTO2010
2010 IFH=1THENH=0 PRINT"[CLR]" GOTO9
2020 GOTO10
2060 K=KX POKEWF(2),16 G=7 S0=HF(1) POKEWF(1),129 FORI=1TO10 REM*CANON 'DADO'
2070 POKEPK+K,G POKES0,1 S=SIN(I) POKEPK-40+K,G POKES0,61
2075 POKEPK-81+K,G
2076 POKES0,121 S=SIN(I) S=SIN(I) IFG=42THENG=32 GOTO2078
2077 G=42 K=0
2078 NEXTI POKES0,1 FORI=1TO100 NEXT POKEWF(1),128 GOTO2004 REM*SONIDO 'OFF'
2087 G=42 K=0
3000 PRINT"[HOM][2CRSRD][RVSON][RED]SC="H1"[HOM][2CRSRD][11CRSRR][PUR]SC="H2 REM
*FIJAR MONTE
3003 R=INT(6*RND(1)+1) FORY=0TOR-1 FORX=0TO5 POKEFNCL(0)+KX,5
3005 POKEFNCL(0),224 NEXTX NEXTY
3007 Y1=Y X1=INT(4*RND(1)+1) X=X1 POKEFNCL(0)+KX,2 POKEFNCL(0),223
3010 FORY=0TO5 FORX=6TO15 POKEFNCL(0)+KX,5
3020 POKEFNCL(0),160 NEXTX NEXTY R=INT(6*RND(1)+1) FORY=0TOR-1 FORX=16TO21
3030 POKEFNCL(0)+KX,5 POKEFNCL(0),224 NEXTX NEXTY
3035 Y2=Y X2=INT(3*RND(1)+17) X=X2 POKEFNCL(0)+KX,4 POKEFNCL(0),233
3040 P1=INT(3*RND(1)+1) P1=P1+5 P2=INT(3*RND(1)+1) P2=16-P2 I=INT(2*RND(1)+1)
3050 IFI=1THENL1=255 GOTO3060
3055 L1=254
3060 I=INT(2*RND(1)+1) IFI=1THENL2=97 GOTO3070
3065 L2=252
3070 X=P1 Y=6 POKEFNCL(0)+KX,5 POKEFNCL(0),L1 FORX=P1+1TOP2-1 Y=6
3072 POKEFNCL(0)+KX,5
3075 POKEFNCL(0),160 NEXTX X=P2 POKEFNCL(0)+KX,5 POKEFNCL(0),L2
3080 FORY=7TO17 RR=INT(6*RND(1)+1)
3085 IFRR=1THENA1=P1+2 L1=254
3090 IFRR=2THENA1=P1+1 L1=254
3095 IFRR>2THENA1=P1 L1=225
3100 RR=INT(6*RND(1)+1) IFRR=1THENA2=P2-2 L2=252
3110 IFRR=2THENA2=P2-1 L2=252
3120 IFRR>2THENA2=P2 L2=97
3130 IFA2<A1ORA1=A2THEN3160
3140 X=A1 POKEFNCL(0)+KX,5 POKEFNCL(0),L1 FORX=A1+1TOA2-1 POKEFNCL(0)+KX,5
3145 POKEFNCL(0),160 NEXTX X=A2 POKEFNCL(0)+KX,5 POKEFNCL(0),L2
3150 P1=A1 P2=A2 IFTT=1THEN3160
3155 NEXTY
3160 PX=P1 PY=Y RETURN
3200 IFR=1THENRR=-(RR) REM*OBTENER VIENTO Y REPRESENTARLO
3210 W=W+(RR/INT(6*RND(1)+4))
3220 IFW<-2THENW=-2
```

más bajo) emiten sonidos musicales y el último registro emite ruidos.

Sonido "On" — Sonido "Off"

El procedimiento para producir sonido en el C-64 es el siguiente:

—Se hace un "Poke" en el registro de volumen al nivel del VIC.

—Restar 127 del nivel del registro de frecuencia del VIC y multiplicar el resultado por dos.

—Se hace un "Poke" en el registro de alta frecuencia al valor obtenido del último paso.

—Se hace un "Poke" en el registro de forma de ondas a 17 o a 129 para encender el sonido correcto.

Una simple operación de "Poke" de la forma en ondas, 16 ó 128, apaga el sonido del C-64. El sonido se apaga al fijar la frecuencia a un valor menor de 128.

La duración del sonido en el C-64 depende en parte si el "Poke" de forma de onda del C-64 se encuentra dentro del circuito de sonido. También podría ser necesario modificar el tamaño del paso de un circuito dadas las diferencias del alcance de la frecuencia del "Poke". La Línea 65 del programa del C-64 constituye un ejemplo de este tipo de cambio.

Línea 27: Yo añadí esta sentencia a la versión del C-64 para que almacenara el ángulo y la pólvora seleccionados en el "array" elegido del jugador. El programa convierte el ángulo a radianes (vea los comentarios en la línea 24).

(Para los numerosos lectores que nunca lo supieron o que olvidaron los ángulos se pueden medir por radianes en vez de por grados. Un radián es la medida de un ángulo —con el vértice en el centro de un círculo— cuyos lados cortan un arco de dicho círculo que es igual en longitud al radio del círculo. Así, la circunferencia total de un círculo —360 grados— es de 2 radianes.)

Líneas 28 a 35: Estas sentencias se añaden para representar el ángulo, la pólvora y la velocidad del viento actuales para ambos jugadores en el campo adicional de la pantalla del C-64. Si estos parámetros no se imprimieran, el lado derecho de la pantalla del C-64 quedaría en blanco. Se tienen que tomar decisiones respecto a esta

Listado 3 (Continuación).

```
3225 IFW>2THENW=2
3235 WD=240 WH=INT(ABS(W*8)) I'WH>15THENWH=15
3240 IFWD0THEN3270
3260 X=PX Y=PY POKEFNCL(0)+KX,6 POKEFNCL(0),103 Y=Y+1 POKEFNCL(0)+KX,6
3261 POKEFNCL(0),103 POKEFNCL(0)+KX+1,6 POKEFNCL(0)+1,102 GOTO3300
3270 X=PX Y=PY POKEFNCL(0)+KX,6 POKEFNCL(0),103 Y=Y+1 POKEFNCL(0)+KX,6
3271 POKEFNCL(0),102  POKEFNCL(0)+KX+1,6 POKEFNCL(0)+1,32
3300 PRINT"[HOM][22CRSRD]"SPC(5)"[RVSON][GRN][12SPC]";
3301 PRINT"[HOM][2CRSRD][RVSON][RED]SC="H1"[HOM][2CRSRD][11CRSRR][PUR]SC="H2
3302 PRINT"[HOM][22CRSRD]"SPC(5)"[RVSON][GRN]VIENTO"INT(ABS(W*80))"KMH[BLK]"; RE
TURN
3500 REM*INICIALIZAR SONIDO,BORDE,FONDO Y CURSOR******
3510 LC=54272 VL=LC+24 BO=53280 BA=BO+1 CU=646
3520 FORNC=LCTOVL POKEMC,0 NEXT
3530 LF(1)=LC HF(1)=LC+1 WF(1)=LC+4
3540 AD(1)=LC+5 SR(1)=LC+6
3550 FORI=2TO3 LF(I)=LF(I-1)+7
3560 HF(I)=HF(I-1)+7 WF(I)=WF(I-1)+7
3570 AD(I)=AD(I-1)+7 SR(I)=SR(I-1)+7 NEXTI
3600 REM*FIJAR SONIDOS VOZ1=RUIDO VOZ2=PROYECTIL**********************
3610 POKEAD(1),15 POKESR(1),133 POKELF(1),200
3620 POKEAD(2),7 POKESR(2),133 POKELF(2),128
3630 AS(1)=45 VS(1)=20 ASS(2)=45 VS(2)=20 REM*FIJAR ANGULO INICIAL Y POLVORA
3640 RETURN
4000 PRINT"[CLR][CRSRD][14SPC]ARTILLERIA[SPC]64[2CRSRD]":REM*INSTRUCCIONES DEL J
UEGO
4010 PRINT"[5SPC]TECLA[11SPC]FUNCION[CRSRD]"
4020 PRINT"[7SPC]F1[12SPC]INCREMENTA[SPC]ANGULO[CRSRD]"
4030 PRINT"[7SPC]F3[12SPC]DECREMENTA[SPC]ANGULO[CRSRD]"
4040 PRINT"[7SPC]F5[12SPC]INCREMENTA[SPC]POLVORA[CRSRD]"
4050 PRINT"[7SPC]F7[12SPC]DECREMENTA[SPC]POLVORA[CRSRD]"
4060 PRINT"[5SPC]RETURN[10SPC]DISPARA[SPC]PROYECTIL[CRSRD]"
4070 PRINT"[7SPC]PULSE[SPC][RVSON]ESPACIO[RVSOFF][SPC]PARA[SPC]COMENZAR"
4080 IFPEEK(197)<>60THEN4080
4090 PRINT"[CLR]"
4100 RETURN
```

Líneas 110, 2060, 3003, 3007, 3010, 3030, 3035, 3070-3075, 3140-3145, sección de la pantalla por cada operación de conversión.

A veces esta zona se debe de usar para representar la puntuación, las instrucciones del juego u otro tipo de información que los jugadores tendrían que apuntar en trozos de papel en la versión del VIC. En otras ocasiones el tamaño del campo de juego se tiene que ampliar. Si es posible es mejor utilizar esta zona porque, si no se hace así, la pantalla da la impresión de ser incompleta.

Líneas 37 y 38: Estas modificaciones son necesarias para aumentar el tamaño del campo en blanco utilizado para que quepan las líneas más largas del C-64. Cuando se dispara el cañón, estas sentencias borran la puntuación y las variables seleccionadas.

3260, 3261, 3270 y 3271: En estas líneas se sustituye el "offset" entre las posiciones de los caracteres y la memoria en colores del C-64 por el "offset" del VIC (30720). El "offset" del C-64 (54272) se almacena como la variable KX en la sentencia 5.

Línea 115: Esta sentencia comprueba que un "Poke" válido será ejecutado en la sentencia 120. Las modificaciones son necesarias dados los diferentes alcances de las frecuencias de sonido del "Poke" de las dos máquinas.

Líneas 1000 y 1110: Se sustituye el número de columnas en la pantalla del C-64 (40) por el número de columnas del VIC (22) en ambas sentencias "Peek".

Líneas 2070 y 2075: Los "Poke" de pantalla para la versión del C-64 se modifican para reflejar el mayor número de columnas. El 22 del VIC se convierte en 40. De modo semejante, el 43 del VIC se convierte en 79, siendo uno menos que dos veces el número de columnas en la pantalla. Finalmente, el VIC 45 se convierte en 81, siendo uno más que dos veces el número de columnas.

Resumen

Este ejercicio demuestra que no es difícil convertir un programa VIC para que se procese en un C-64. Dado que las dos máquinas comparten el mismo lenguaje Basic (Commodore 2.0), todas las sentencias lógicas, y las expresiones matemáticas son totalmente transferibles. Principalmente los programas se tienen que modificar cuando entran en juego los "Peek" y los "Poke". La Tabla 3 presenta algunas direcciones importantes que normalmente se encuentran en los listados del VIC, con las direcciones correspondientes para el C-64.

Si vd. tropieza con estas direcciones en el proceso de conversión, cambie las direcciones de la forma que hemos mencionado antes, y obtendrá los resultados correctos del programa. Si tiene alguna dificultad o encuentra una dirección no mencionada, la "Guía de Referencia para Programadores del Commodore" le proporcionará una información adicional. Si sigue teniendo problemas, un grupo de usuarios o amigos podrán ayudarle.

VENTANA CBM (IV)

Los capítulos anteriores fueron publicados en Commodore Club. Aquellos lectores que los deseen, les rogamos nos lo comuniquen para poder enviarles copia de dichos artículos.



READY
(del chip a la base de datos) (I)

Cumpliendo con lo prometido en números anteriores, voy a comentaros en éste algunos aspectos del sistema co-residente MEC/DOS (MicroElectrónica y Control/Disc Operating Sistem).

MEC/DOS es un sistema operativo co-residente con el BASIC y el KERNAL, escrito en assembler por MEC SOFT para los micros de la serie 7XX de COMMODORE.

Por Rafael NAVARRO

Recomendado para C-64 y C-700

Lo primero que cabe destacar en él es que su nombre no lo define totalmente ya que los criterios considerados para su bautizo fueron más comerciales y estéticos que técnicos; no se trata exactamente de un sistema operativo de disco ya que, por un lado, funciones elementales de sistema, como la copia de discos o su formatación se dejan al sistema base de COMMODORE y, por otro, MEC/DOS abarca también otras áreas, como la edición de pantallas y el cálculo.

Desde este punto de vista, podría decirse que MEC/DOS es un conjunto de ayudas a la programación que, sin perder su carácter universal, se enfoca a facilitar la programación y el desarrollo de aplicaciones para gestión y control de actividades en la empresa. Acordes con la filosofía de COMMODORE, hemos integrado el sistema en un cartucho con 16K de eprom, conteniendo el código y 8K de ram de uso interno, evitando de este modo los tiempos de carga (bootstraps) que otros sistemas padecen. Se trata pues de un concepto nuevo entre los productos desarrollados tanto para COMMODORE como para otros sistemas:

—no ocupa RAM del usuario.
—no interfiere el BASIC residente aunque aprovecha algunas de sus rutinas
—debido a su alto nivel de estructuración interna es independiente del lenguaje empleado
—para cuando se trabaje con el BASIC residente (interpretado) dispone de su propio interpretador que trabaja en base a comandos con nombre propio (nada de SYS 34567 o CALL 56678 totalmente carentes de significado) y opera en entrada/salida con parámetros (variables) bajo el mismo formato que el BASIC
—es extremadamente fácil de entender, aprender y utilizar
—dispone de un sistema automático de protección contra errores tanto de usuario como de sistema, lo cual le confiere un alto nivel de seguridad e información, ya que no sólo lo avisa de los estados de error si no que informa de las características de los mismos mediante completos mensajes.

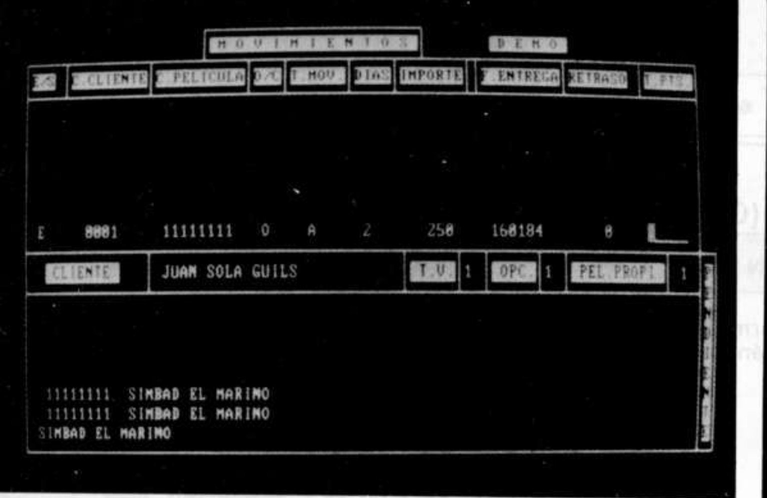
MEC/DOS abarca las siguientes áreas de trabajo:

—Tratamiento de ficheros relativos.
—Tratamiento de ficheros de claves.
—Tratamiento de ficheros tablas.
—Tratamiento de campos (des/empaquetado)
—Edición y tratamiento de pantallas.
—Múltiple precisión.
—Intérprete MEC/DOS (con BASIC residente).

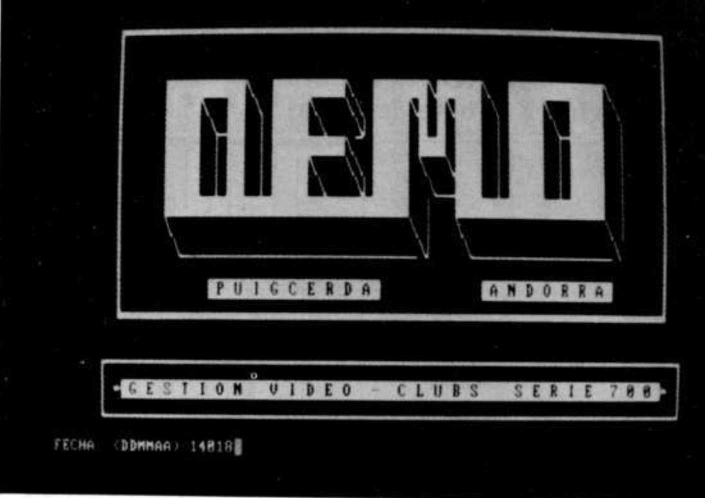
Al final de este artículo se encuentra la tabla resumen de los comandos que acompaña al manual de MEC/DOS. Considérese que en ella aparecen los comandos por el nombre que es inteligible para el interpretador MEC/DOS. Cuando se opere bajo un entorno que no sea el interpretador BASIC de COMMODORE (compilador BASIC, —PET SPEED—, PASCAL —DTL—, u otros) el formato de utilización será diferente, pero siempre posible.

El interpretador

El interpretador está constituido por una serie de programas que comunican al programador con el núcleo de MEC/DOS. Tanto desde el punto de vista interno como en su relación con el programador, MEC/DOS utiliza un buffer



Base de datos



Base de datos

(Fotos cedidas por DENO)

(área de memoria reservada) para comunicar datos de entrada/salida entre las diferentes rutinas que constituyen el sistema y entre éstas y el programador. La función del interpretador es facilitar dicha comunicación a través de variables BASIC. De modo sucinto, podemos definir el siguiente **modus operandi** para el interpretador:

—lectura del comando
—comprobación de la sintaxis
—lectura de parámetros y formatación de los mismos
—ejecución del comando propiamente dicho
—recogida de los datos de salida
—formatación de los mismos
—devolución de resultados a través de variables BASIC
—actualización de la variable OK (avisos y errores)
—impresión de mensaje de error y listado de la línea donde se produjo (opcional)

MEC/DOS ofrece 50 comandos diferentes cuya longitud es de 6 caracteres reconocibles, tras los cuales puede añadirse cualquier afijo para hacerlos más significativos todavía. Por ejemplo, el comando "LIBERA" (liberación de un registro de fichero relativo para su posterior utilización tras una baja) puede ser escrito como "LIBERA EL REGISTRO QUE A CONTINUACION TE INDICO" siendo el aspecto final de la instrucción algo así como: "LIBERA EL REGISTRO QUE TE INDICO", NF, NR, para liberar al registro NR del fichero NF. Como se ve, no puede ser más claro. En este caso, solamente existen dos parámetros de entrada (NF y NR) y ninguno de salida y obviamos que OK —que será cargado con un 0 si la operación se ha llevado a cabo sin error y con otro valor en caso contrario— no es un parámetro de salida si no la variable de estado de MEC/DOS. El sistema permite la utilización de los siguientes parámetros.

En entrada:
—constantes numéricas
—constantes alfa
—variables numéricas simples o array
—variables alfa simples o array
—cualquier tipo de expresión alfa o numérica.

En salida:
—variables numéricas simples o array
—variables alfa simples o array

Aunque es imposible en este espacio dar una idea exacta de lo que representa MEC/DOS en todos los aspectos, las siguientes líneas pretenden dar una idea muy general de sus posibilidades.

Ficheros relativos

Los ficheros relativos son ficheros de estructura muy similar a los del BASIC de COMMODORE y otros de los basics que ruedan bajo CP/M o MS/DOS. De hecho, tanto los RELATIVOS como las CLAVES DE MEC/DOS están soportados por ficheros de estructura RELATIVO de COMMODORE. Las diferencias fundamentales estriban en los siguientes puntos.

—MEC/DOS permite tener abiertos 12 ficheros simultáneos
—se efectúa un control de los registros libres y ocupados
—se gestionan automáticamente los campos de cada registro procediendo a su empaquetado y desempaquetado en los procesos de entrada/salida.
—a tal efecto no es necesaria ninguna preparación previa en el programa (existe un descriptor, generado en el momento de definir el fichero). Basta con definir los ficheros una sola vez mediante un utilitario y MEC/DOS recuerda siempre las características de los mismos
—existe una detección automática de errores que no obliga al testeado constante de variables de errores (DS y DS$ en COMMODORE, ER, EL, ER$ en otros sistemas)
—en el capítulo de seguridad, cabe destacar además, la existencia de comandos de repaso de seguridad y arreglo automático de posibles errores.
—grandes facilidades para la ampliación y/o cambio de estructura de los ficheros.

TABLA RESUMEN DE COMANDOS DEL MEC/DOS

INICIALIZACION

COMANDO	EXPLICACION	VALORES DE OK
INIT	Inicializa MEC/DOS reseteando las unidades de disco conectadas. Pone como periférico por defecto de unidad de discos 8.	OK = 0; correcto
DISK,PE	Cambia el número de periférico por defecto para los comandos CHARGE y ASSIGN.	OK = 0; correcto

FICHEROS (GENERAL)

COMANDO	EXPLICACION	VALORES DE OK
ASSIGN, nf, nf$ [,d(dr)] [,u(pe)] [,P]	Asigna el fichero nf permitiendo al operador trabajar refiriéndose solamente a nf.	OK = 0; correcto OK = 1; comando no permitido OK = 2; nf ya asignado OK = 3; demasiados ficheros asignados OK = 4; demasiados ficheros prioritarios OK = 13; error de checksum en carga de tablas.
KEEP,nf	Para ficheros de tablas graba la tabla y reabre el fichero. Para ficheros relativos graba el descriptor si ha cambiado el primer registro libre y reabre el fichero. En ficheros de claves, cierra y abre el fichero.	OK = 0; correcto OK = 5; fichero no asignado
LOCK,nf [nf,nf...]	Cierra ficheros y graba, si procede, descriptores y/o tablas.	OK = 0; correcto OK = 5; fichero no asignado

FICHEROS RELATIVOS

COMANDO	EXPLICACION	VALORES DE OK
USE,NF,NR	Coloca el registro NR del fichero NF en estado de "ocupado", permitiendo de este modo, grabar (WRITE) sobre el mismo.	OK = 0; correcto OK = 5; NF no asignado OK = 6; NR fuera de rango OK = 9; NR ya ocupado
LIBERATE,NF,NR	Libera el registro NR del fichero NF.	OK = 0; correcto OK = 1; comando no apropiado OK = 5; NF no asignado OK = 6; NR fuera de rango OK = 7; registro ya liberado
READ,NF,NR	Carga en el buffer el registro NR del fichero NF, dando acceso a consultas y variaciones del contenido de NR.	OK = 0; correcto OK = 5; NF no asignado OK = 6; NR fuera de rango OK = 7; registro libre
WRITE, NF,NR	Graba el contenido del buffer del fichero NF sobre el registro NR del mismo. El contenido del buffer permanece intacto. El registro debe estar ya ocupado.	OK = 0; correcto OK = 5; NF no asignado OK = 6; NR fuera de rango OK = 7; registro libre
AVAIL,NF,NR	Devuelve en OK la disponibilidad del registro NR del fichero NF. Los valores de OK 7 y 9 no producen mensaje en este comando, por meramente informativos.	OK = 5; NF no asignado OK = 6; (NR fuera de rango) OK = 7; registro libre OK = 9; registro ocupado
FIRST,NF,VR	Devuelve en VR el valor del primer registro libre del fichero NF. Cero si NF esta lleno.	OK = 0; correcto OK = 5; NF no asignado OK = 8; NF completo
RELINK,NF	Repasa, caso de existir sospechas sobre su integridad, las cadenas de control de registros libres del fichero NF.	OK = 0; correcto OK = 1; comando no apropiado

(Continuará en el número 3.)

¡Arriba Periscopio!

Los gráficos, la generación de sonido, las comunicaciones y un número de funciones de la configuración de un equipo dependen de que el usuario seleccione unos bits específicos de una posición definida de memoria, que haga un "PEEK" en dicha posición para saber cuáles son los bits puestos a 1 ó a 0 y luego haga un "POKE" en esa posición de memoria para poner a 1 o a 0 los bits que necesite.

COMMODORE 64
o
VIC-20 de
3 K o más

Para nosotros, quienes a veces tenemos problemas para pensar en valores hexadecimales y binarios e ignoramos cuáles son los valores almacenados en la memoria, el programa "Periscopio" para el Commodore 64 y el VIC-20 proporciona un instrumento útil para que entendamos mejor nuestras máquinas.

El programa "Periscopio" se escribe totalmente en Basic y permite que el usuario vuelque los valores hexadecimales almacenados en una serie de posiciones de memoria, corregir (poner un PATCH) (en hexadecimal) unos valores en una serie de posiciones de memoria, desensamblar los valores hexadecimales de una serie de posiciones de memoria en instrucciones de lenguaje ensamblador del 6502/6510, y convertir los valores de números de decimal, hexadecimal o binario en todas las tres bases.

Operación del Programa.

Cuando se ejecuta el programa "Periscopio", lo primero que hace es borrar la pantalla y representar un menú de opciones. Se selecciona una opción mediante la pulsación de un número de 1 a 5 antes de pulsar la tecla de entrar (ENTER). Cuando se haya seleccionado una opción correcta, el programa pregunta o bien una dirección de principio (¿Sí?) o bien una pregunta de conversión de base (13=DEC 2=HEX 3=BIN)?

Vamos a estudiar las opciones para ver lo que pueden hacer para nosotros.

Opción 1: Volcado (Dump) - La opción de volcado representa los valores almacenados en la memoria en notación hexadecimal (dos dígitos hexadecimales para cada byte de memoria). Los valores de memoria se representan a partir de una dirección inicial a una dirección final, introducidos mediante el teclado. Cuando se selecciona la opción, se representa una pregunta de dirección de principio (¿Sí?)

A continuación el programa espera que se introduzca una dirección de memoria de dos bytes (cuatro dígitos hexadecimales). Si se produce un error al introducir la dirección, no se puede corregir mediante una operación de retroceso (backspace), dado que el programa no reconoce INST/-DEL. Sin embargo, los datos de la dirección se pueden volver a teclear justo después del error en la misma línea. El programa acepta los últimos cuatro dígitos tecleados antes de haber pulsado la tecla "RETURN", sea cual sea el número total de caracteres que hayan sido tecleados en la línea.

Después de introducir la dirección inicial, el programa contesta con un mensaje de dirección final de >. La dirección final se introduce de la misma forma que la dirección inicial, considerando los últimos cuatro dígitos tecleados como el valor real.

Si se introducen solamente dos o tres dígitos hexadecimales, el programa supone que los caracteres de la extrema izquierda son ceros. Por ejemplo, si se teclea AE hexadecimal y se pulsa la tecla "RETURN", el programa supone que significa la dirección de OOAE hexadecimal.

Si el programa tiene las direcciones de principio y final para el volcado de memoria, se presenta un mensaje de "Salida de Impresora (Sí/No). Una respuesta de Sí dirige la salida a la impresora 1525 de Commodore. Si se introduce una N, o simplemente se pulsa la tecla "RETURN", la salida se dirige a la pantalla.

Si se pulsa la tecla "ENTER", el programa imprime una dirección de memoria hexadecimal seguida por ocho valores hexadecimales almacenados en la memoria del VIC-20. El programa sigue imprimiendo las direcciones de memoria, seguidas por los valores de memoria en grupos de seis u ocho bytes, hasta que se llega a la dirección final o hasta que se pulsa cualquier tecla alfanumérica en el teclado cuando se representan los valores en la pantalla.

Cuando se pulsa una tecla durante una operación de volcado en la pantalla, la salida se detiene y el programa espera que se pulse otra tecla antes de continuar la operación. Así los datos no se pierden fuera de la pantalla antes de que se puedan leer. Si la tecla que se pulsa para interrumpir la representación es la "X", la operación de volcado finaliza y el programa vuelve a presentar la pregunta ¿Yes? (¿Sí?). Si la tecla "X" se vuelve a pulsar, el programa vuelve al menú.

Si se introduce la dirección hexadecimal, el programa pedirá la dirección final y la salida de impresora, y luego realiza una operación de volcado de los valores de memoria empezando por la nueva dirección inicial. Cuando se llega a la dirección final, el programa representa la pregunta ¿Yes? y otro volcado puede ser seleccionado, o si no, se pulsa la tecla "X" para volver al menú.

Opción 2: Corrección (Patch) - La opción de corrección (Patch) permite introducir valores hexadecimales en la memoria. Cuando la opción es seleccionada, el programa presenta la segunda Yes? Una dirección inicial debe introducirse de la misma manera en que se hizo con la opción de volcado, o si no, se puede pulsar la tecla "X" para volver al menú.

Después de introducir una dirección inicial, el programa presenta el mensaje >.



Se introduce la dirección final o una X para volver al menú. Después de introducir la dirección final, el programa presenta la dirección inicial en hexadecimal y espera que se introduzca un valor hexadecimal de un byte. Un error al teclear se puede corregir de la misma forma en que se hace cuando se introduce una dirección; se teclea el valor correcto justo después del error. El programa acepta los últimos dos dígitos hexadecimales en la línea como el valor correcto que se coloca en la memoria.

El programa continuará representando la siguiente posición de memoria y esperará que el usuario entre un nuevo valor. Esta secuencia continuará hasta que la posición final sea alcanzada o hasta que se pulse la tecla "X". Si la tecla "X" es pulsada, la rutina de entrada de datos terminará y aparecerá el mensaje "¿Sí?". Lo mismo que en el caso de la función de volcado, un nuevo juego de direcciones pueden entrarse, o pulsando la tecla "X" otra vez podemos volver al menú.

Opción 3: Desensamblado - La operación de desensamblado presentará una sección de memoria como instrucciones del ensamblador del 6502/6510. La rutina traducirá los valores hexadecimales en una posición de memoria particular de 6502/6510 en mnemónicos del lenguaje "assembler" del 6502/6510 para una instrucción particular, y la presenta en pantalla. La rutina a continuación presentará los valores de memoria de la siguiente o de las siguientes dos posiciones, si esto lo requiere el direccionamiento del programa en lenguaje máquina.

Los mensajes de la opción desensamblador y la entrada de datos son exactamente iguales que en el caso de la opción de volcado. Se presentan los mensajes de entrada y final de las direcciones y el mensaje de salida e impresora. En ambos mensajes, la tecla "X" puede ser pulsada para volver al menú. Mientras que el desensamblaje se presenta en la pantalla, la rutina puede ser interrumpida de la misma forma que la rutina de volcado. Se puede especificar una nueva sección de memoria o volver al menú.

Opción 4: Conversión - La opción de conversión puede convertir un valor numérico entrado como decimal, hexadecimal o binario en cualquiera de las dos bases restantes y su valor presentado en pantalla. Cuando se selecciona esta opción, el programa presentará el mensaje de conversión (1=DEC 2=HEX 3=BIN?). A continuación, hay que pulsar un número correspondiente a la base que se desee a bien pulsar la tecla "X" para volver al menú. Después de haber seleccionado la base, el programa presentará el mensaje DEC?, HEX? o BIN?. A continuación el programa esperará la entrada del valor en la base especificada.

Los números decimales tienen que estar comprendidos entre 0 y 65535 sin ningún espacio entre los dígitos. Los números hexadecimales pueden tener hasta cuatro caracteres de longitud y tienen que introducirse de la misma forma que las entradas de direcciones. Los números binarios tienen que tener desde uno, hasta máximo 16 dígitos, formados por unos y ceros. De la misma forma que con los números decimales, es un error introducir espacios en blanco entre los distintos dígitos. Cuando se estén introduciendo números decimales o binarios, se puede utilizar la tecla de "borrado" para volver a la posición anterior y corregir la última entrada antes de pulsar la tecla "ENTER".

Cuando un valor ha sido introducido, el programa presentará su valor convertido en una de las otras dos bases soportadas. Un número decimal se representará como un valor entero. Un número hexadecimal se presentará como cuatro caracteres hexadecimales. Un número binario se representará como dos grups de ocho dígitos (que es la representación binaria de dos bytes).

Después de haber presentado los valores convertidos, el programa presentará el mensaje (1=DEC 2=HEX 3=BIN)? y otro valor de base puede ser introducido, o se puede pulsar la tecla "X" para volver al menú.

Opción 5: Fin - La opción fin comprueba para asegurarse de que la impresora está desconectada y que el programa vuelve a dar control al editor Basic de pantalla del Commodore.

Descripción del Programa

El programa "Periscopio" ha sido diseñado para ofrecer un instrumento útil y al mismo tiempo utilizar la mayor memoria posible. Con esto en mente, en cada línea se han escrito el mayor número posible de instrucciones. No se han utilizado las instrucciones REM por el mismo motivo. Cuando se introduce el programa, las líneas 600-610 tienen que ser descartadas para ahorrar espacio. El programa ocupará entonces aproximadamente 3000 bytes de memoria y aproximadamente 3785 bytes de memoria cuando funcione.

El programa puede ser condensado posteriormente usando los comandos abreviados de dos teclas soportados por el Basic de Commodore. Por ejemplo, las líneas 611 y 615 pueden ser combinadas. Los comandos de datos de las líneas 783-798 pueden ser condensados en menos líneas en total. El programa ha sido escrito tal cual para facilitar la entrada del listado dado que las líneas han sido construidas con los comandos abreviados de dos teclas que normalmente exceden de los 80 caracteres en el Commodore 64 y de los 88 caracteres en el VIC-20 cuando se listan en pantalla. Aunque las líneas traducidas entren dentro de una línea de Basic en memoria, esta línea no puede ser editada en la pantalla sin perder datos, de forma que la línea tiene que ser escrita de nuevo desde cero para corregir los errores. El programa tiene que ser escrito como en el listado, hay que ejecutarlo para detectar errores, y corregirlo. Una vez que funcione correctamente, puede ser condensado.

El listado del programa puede ser utilizado tal cual en el Commodore 64. Hay que cambiar una línea para ejecutar el programa en el VIC-20. En la línea 900, el valor (P-S)/8 aparece dos veces. Cambiar ocho por seis en ambos sitios, y la opción de volcado entrará en la anchura del VIC-20.

En un VIC-20 no ampliado, el usuario podría querer borrar algunas opciones para tener más memoria libre cuando se ejecuta el programa "Periscopio". Por ejemplo, borrando las líneas 670-696 y borrando el valor 670 desde la línea 655 y PRINT"4) CONVERT:" desde 640 y cambiando PRINT"5) END en PRINT"4) END", la opción entera de conversión será eliminada del programa. Como herramienta para modificaciones posteriores del programa, a continuación tenemos unas descripciones de las rutinas.

Líneas 611-620: Esta rutina inicializa y llena dos "arrays" con datos. El "array" 0% contiene los valores empaquetados que, cuando son desempaquetados, nos dan el puntero del "array" M$ y el puntero de las subrutinas correspondientes para imprimir los diferentes modos del direccionamiento del 6502/6510.

Líneas 630-659: Estas líneas contienen las rutinas del programa principal, que presentan el menú de opciones en las líneas 630 y 640, hacen un "input" de la opción seleccionada en la línea 645 y resetean la impresora y las direcciones de las entradas de las subrutinas en la línea 650. La línea 655 dirige el programa a la subrutina de la opción seleccionada.

Líneas 665-667: Esta rutina es llamada para inhibir la salida a la impresora.

Línea 660: Esta línea llama la rutina que inhibe la impresora y a continuación acaba el programa.

Líneas 670-696: La opción de rutina de conversión imprime los mensajes de conversión en la línea 670. Las líneas 672-682 tienen las instrucciones de entrada de selección de la base, e imprimen el número de bases y llaman las subrutinas de conversión. Las líneas 684-686 convierten una entrada binaria en un número decimal y luego llaman la rutina de impresión del número decimal en la línea 688.

Un número decimal es convertido en binario, y luego impreso como un número binario de 16 bits en las líneas 690-696. La línea 696 llama una subrutina en la línea 950; esta subrutina convierte un número

Listado 1

```
UP PERISCOPE

600 REM********************************
601 REM*                              *
602 REM*        # PERISCOPE #         *
603 REM* FOR COMMODORE-64 AND VIC 20 *
604 REM*                              *
605 REM*                              *
606 REM*                              *
607 REM*                              *
608 REM*                              *
609 REM*                              *
610 REM********************************
611 DIMM$(56),O%(255):FORP=ØTO56:READM$(P):NEXT:FORP=ØTO255:READW
615 IFW<99THENW=W*100
620 O%(P)=W:NEXT
630 PRINTCHR$(147):R=Ø:PRINTCHR$(18);"*[SPC]PERISCOPIO[SPC]AQUI![SPC]*":PRINT:PR
INT"1)[SPC]VOLCADO"
640 PRINT"2)[SPC]CORRECCION":PRINT"3)[SPC]DESENSAMBLADO":PRINT"4)[SPC]CONVERSION
":PRINT"5)[SPC]FIN"
645 PRINT:INPUT"OPCION[SPC]";C:IFC>3ORC<1THEN655
650 GOSUB665:GOSUB970:IFRTHEN630
655 ONCGOSUB900,800,700,670,660:R=Ø:IFC>3THEN630
659 GOTO650
660 :GOSUB665:END
665 IFL=1THENPRINT#4:CLOSE4:L=Ø:P$=""
667 RETURN
670 R=Ø:PRINT"(1=DEC[SPC]2=HEX[SPC]3=BIN)[SPC]?"
672 GETK$:IFK$=""THEN672
674 IFK$="X"THENRETURN
676 IFK$="1"THENPRINT"DEC";:E=Ø:INPUTE:P=E:GOSUB696:GOSUB690
678 IFK$="3"THENPRINT"BIN";:GOSUB684:GOSUB696
680 IFK$="2"THENPRINT"HEX?";:GOSUB980:IFR<1THENGOSUB688:GOSUB690
682 PRINT:GOTO670
684 E=Ø:Q=Ø:K$="Ø":INPUTK$:FORX=LEN(K$)TO1STEP-1:Q=Q+1
686 E=E+VAL(MID$(K$,Q,1))*2↑(X-1):NEXTX:GOSUB688:RETURN
688 PRINT"DEC=";E:RETURN
690 PRINT"BIN=";:FORX=15TO1STEP-1:Q=Ø:IFE-2↑X>=ØTHENE=E-2↑X:Q=1
692 IFX=7THENPRINT"[SPC]";
694 PRINTRIGHT$(STR$(Q),1);:NEXTX:PRINTRIGHT$(STR$(E),1):RETURN
696 P=E:PRINT"HEX=";:GOSUB950:PRINT:RETURN
698 P=E:NEXTP:PRINT:RETURN
700 GOSUB990:FORP=STOE:IFLTHEN710
705 GOSUB895:IFRTHEN698
710 GOSUB950:W=O%(PEEK(P)):Z=INT(W/100):W=W-Z*100:PRINT"[SPC]";M$(Z);"[SPC]";
720 ONWGOSUB725,780,735,740,782,750,755,760,765,770,773,775:PRINT:NEXT:RETURN
725 PRINT"#";:GOTO780
735 GOSUB780:PRINT",X";:RETURN
740 GOSUB780:PRINT",Y";:RETURN
750 GOSUB782:PRINT",X";:RETURN
755 GOSUB782:PRINT",Y";:RETURN
760 P=P+1:Q=P:H=PEEK(P):IFH>127THENP=P-256
763 P=P+1+H:PRINT"$";:GOSUB950:P=Q:RETURN
765 PRINT"(";:GOSUB780:PRINT",X)";:RETURN
770 PRINT"(";:GOSUB780:PRINT"),Y";:RETURN
773 PRINT"A";:RETURN
775 PRINT"(";:GOSUB782:PRINT")";:RETURN
780 PRINT"$";:P=P+1:H=PEEK(P):GOTO920
782 PRINT"$";:P=P+2:H=PEEK(P):GOSUB920:H=PEEK(P-1):GOTO920
783 DATA???,ADC,AND,ASL,CMP,EOR,LDA,ORA,SBC,STA,BCC,BCS,BEQ,BIT,BMI,BNE,BPL,BRK
784 DATABVC,BVS,CLC,CLD,CLI,CLV,CPX,CPY,DEC,DEX,DEY,INC,INX,INY,JMP,JSR,LDX,LDY
785 DATALSR,NOP,PHA,PHP,PLA,PLP,ROL,ROR,RTI,RTS,SEC,SED,SEI,STX,STY,TAX,TAY
786 DATATSX,TXA,TXS,TYA,17,709,,,,702,302,,39,701
787 DATA311,,,705,305,,1608,710,,,,703,303,,20,707,,,,706,306,,3305,209,,,1302
788 DATA202,4202,,41,201,4211,,1305,205,4205,,1408,210,,,,203
789 DATA4203,,46,207,,,,206,4206,,44,509,,,,502,3602,,38,501,3611,,3205,505,3605
790 DATA,1808,510,,,,503,3603,,22,507,,,,506,3606,,45,109,,,,102,4302,,40,101
792 DATA4311,,3212,105,4305,,1908,110,,,,103,4303,,48,107,,,,106,4306,,
793 DATA909,,,5002,902,4902,,28,,54,,5005,905,4905,,1008,910,,,5003,903,4904,,56
794 DATA907,55,,,906,,,3501,609,3401,,3502,602,3402,,52,601,51,,3505,605,3405,
795 DATA1108,610,,,3503,603,3403,,23,607,53,,3506,606,3407,,2501,409,,,2502
796 DATA402,2602,,31,401,27,,2505,405,2605,,1508,410,,,,403,2603,,21,407,,,,406
797 DATA2606,,2401,809,,,2402,802,2902,,30,801,37,,2405,805,2905,,1208
798 DATA810,,,,803,2903,,47,807,,,,806,2906,
800 FORP=STOE:GOSUB950:PRINT"[2SPC]";:H$="":GOSUB820:IFRTHEN698
810 POKEP,H:PRINT:NEXT:RETURN
820 GETK$:IFK$=""THEN820
830 W=ASC(K$):IFW=88THENR=1:RETURN
840 IFW=13ANDLEN(H$)>1THEN870
850 IFW<48OR(W>57ANDW<65)ORW>70THEN820
860 PRINTK$;:H$=H$+K$:GOTO820
870 W=ASC(RIGHT$(H$,2))-48:IFW>9THENW=W-7
880 W=W*16:H=ASC(RIGHT$(H$,1))-48:IFH>9THENH=H-7
890 H=H+W:RETURN
895 GETK$:IFK$=""THENRETURN
897 IFK$="X"THENR=1:RETURN
898 WAIT198,1:POKE198,Ø:RETURN
900 GOSUB990:FORP=STOE:IFINT((P-S)/8)=(P-S)/8THENPRINT:GOSUB950
```

decimal en hexadecimal e imprime los caracteres hexadecimales.

Línea 698: Esta rutina nos da la forma de salirnos de un "loop" For... Next y de volver al menú. Si salimos de esta subrutina sin hacer un "RETURN", o si un "loop" For... Next es acabado demasiado pronto, el registro de "stack" del 6502/6510 se llenará de basura y el resultado es un error de "Out Of Memory" cuando, sin embargo, todavía hay memoria disponible. Esta rutina limpia todos los internos de la máquina cuando salimos del "loop".

Líneas 700-782: Esta rutina contiene la opción de desensamblaje. La línea 700 llama la subrutina de activación de la impresora, y luego inicializa los "loops" For... Next desde el principio hasta el final de las direcciones de memoria. Si la impresora es activada, el programa saltará a la línea 720. En caso contrario, la línea 705 es ejecutada y llamará a una rutina de interrupción en la línea 950 para comprobar una condición de parada de salida.

El programa a continuación cargará el valor desde la memoria y utilizará este valor para apuntar a un elemento del "array" 0%. El elemento del "array" será dividido a continuación por 100 para apuntar dentro del mnemónico dentro del lenguaje "assembler" en el "array" M$. El resultado se utilizará en la línea 720 para dirigir el programa a una subrutina correcta e imprimir el operando en el modo de direccionamiento requerido del 6502/6510. Estas subrutinas están ubicadas desde la línea 725 a la línea 782.

Líneas 783-798: Estas líneas contienen los valores de datos utilizados para cargar el "array" M$ y el "array" 0% cuando el programa se ejecuta por primera vez.

Líneas 800-810: Estas líneas contienen la rutina de la opción de corrección (patch). En la línea 800 empieza un "loop" For... Next desde la dirección de principio hasta la dirección de final. A continuación el programa llama la rutina de conversión decimal a hexadecimal en la línea 950 e imprime la posición de memoria. La subrutina de entrada hexadecimal es llamada en la línea 820, y los datos introducidos se escriben en la posición de memoria en la línea 810.

Líneas 820-890: Esta rutina introduce dos caracteres hexadecimales desde el teclado y convierte su valor en un número decimal. La rutina pone a 1 un "flag" de "RETURN" (R) si una salida de esta rutina es solicitada pulsando la tecla "X".

Líneas 895 # 898: Esta rutina es la subrutina de solicitud de interrupción llamada por las opciones de volcado y desensamblaje para comprobar si ha habido una solicitud de parada de salida.

Líneas 900-910: Esta rutina contiene la opción de volcado. La línea 900 llama la subrutina que activa la impresora, y luego inicializa el "loop" For... Next para la dirección de principio y final. El programa luego comprueba la condición de



Listado 1 (continuación)

```
905 IFL=0THENGOSUB895:IFRTHEN698
910 H=PEEK(P):PRINT"[2SPC]";:GOSUB920:NEXT:PRINT:RETURN
920 W=INT(H/16):H=H-W*16:GOSUB930:W=H
930 W=W+48:IFW>57THENW=W+7
940 PRINTCHR$(W);:RETURN
950 H=INT(P/256):Z=P-H*256:GOSUB920:H=Z:GOTO920
970 PRINT"SI?";:GOSUB980:IFRTHENRETURN
975 S=E:PRINT">";
980 H$="":GOSUB820:IFRTHENRETURN
985 H$="00"+H$:E=H:H$=MID$(H$,LEN(H$)-3,2):GOSUB870:E=E+H*256:PRINT:RETURN
990 L=0:PRINT"SALIDA[SPC]A[SPC]"
991 INPUT"IMPRESORA[SPC](S/N)";P$:IFLEFT$(P$,1)="S"THENL=1:OPEN4,4,0:CMD4
993 RETURN
```

final de pantalla, y luego en la línea 950 llama la rutina de conversión de decimal a hexadecimal. La línea 905 llama una subrutina de solicitud de interrupción en la línea 895 si una salida es presentada en la pantalla, y luego se carga un valor desde una posición de memoria y luego se llama la rutina de conversión decimal a hexadecimal y se imprime con la rutina en la línea 920.

Líneas 920-950: Esta rutina convierte un número decimal en hexadecimal. Si se entra en la línea 950, cuatro caracteres hexadecimales se imprimirán. Si se entra en la lnea 920, se imprimirán dos caracteres hexadecimales.

Líneas 970-975: Esta rutina es la dirección de principio y de final de la rutina de entrada. El mensaje ¿Sí? se imprimirá en la línea 970, y luego el programa llamará en la línea 980 y 985 la rutina de cálculo de conversión de dirección hexadecimal a dirección decimal.

La línea 980 limpia la variable H$, y llama la rutina de entrada del carácter hexadecimal en la línea 820 y luego ejecuta la línea 985 para convertir el carácter hexadecimal a número decimal.

Líneas 990-993: Esta subrutina imprime el mensaje de salida a impresora y activa la impresora cuando se contesta por un Sí desde desde el teclado.

Conclusión

El programa "Periscopio" es una herramienta útil, un monitor simple para entrar y presentar programas escritos en lenguaje máquina. Su finalidad es la de ayudar a los nuevos usuarios del Commodore 64 y del VIC-20 para conocer mejor su máquina. Al mismo tiempo proporciona una forma para entrar programas cortos escritos en código máquina del 6502/6510 en el equipo. Con este programa funcionando, y con la "Guía de Referencia del Programador" para el Commodore 64 o el VIC-20, las formas de trabajar internas de la máquina serán mucho más comprensibles. ■

DISK-O-VIC

Se le contagiará la Fiebre del Sábado Noche con este programa de utilidad que proporciona 13 mandatos relacionados a discos y hará bailar el VIC-20 y la unidad de discos 1541. Haz un hueco, John Travolta.



El Commodore VIC-20 y la unidad de discos 1541 constituyen una combinación muy importante. Por supuesto, el VIC-20 es un ordenador muy completo basado en 6502, que ofrece muchas características profesionales como un juego completo de mandatos Basic, un teclado profesional y opciones para una memoria ampliada.

A menudo, un sistema inicial lleva una unidad de cassette como método de almacenamiento en general, pero a medida que asciende el nivel del usuario y aumenta la necesidad de un E/S más rápido, se hace imprescindible una unidad de discos. Igual que las demás unidades de discos flexibles de Commodore, la unidad 1541 es inteligente. Esto significa que por separado la unidad 1541 es un ordenador capaz de realizar muchas funciones sin la intervención del sistema principal.

De hecho, la 1541 contiene su propio microprocesador, un par de VIAs (adaptadores de interface versátiles), 2K de RAM y un sistema de operaciones completo en ROM. Esto conduce a dos hechos importantes. Primero, dado que el sistema de la 1541 resulta tan completo, no necesita robar RAM del programa del usuario del sistema principal. Al contrario de muchas combinaciones de unidad de discos/ordenador, el VIC-20 dispone de la misma cantidad de espacio de programa con o sin la unidad de discos.

Segundo, dado que la 1541 es inteligente, puede ser programada externamente para que realice muchas funciones útiles. Realmente, la unidad no tiene límites fijos en el sentido de que si una función determinada no existe de antemano dentro del sistema operativo de discos, se puede escribir un programa para generar tal función.

VIC-20

1514 Disk Drive
Editor/assembler o
machine languaje

DISK-O-VIC

Este artículo describe un paquete de utilidad que se llama DISK-O-VIC, que añade trece comandos nue-

vos relacionados con discos al VIC-20. Dichos comandos formarán parte del Basic y pueden ser utilizados en la modalidad inmediata para simplificar las operaciones rutinarias con la unidad de discos. Algunos de los comandos, como DLOAD y DSAVE, son ampliaciones de comandos Basic. Otros, como "Screatch" y "Rename", se utilizan para mantener los discos limpios y en orden. Finalmente, otro grupo añade características como listados de mensajes de error, representación del directorio, etc.

El paquete de utilidad DISK-O-VIC está escrito en lenguaje máquina para que sea más rápido y más flexible. Después de cargarse e inicializarse puede quedarse allí durante toda una sesión de programación. Debido a la característica especial de carga (detallada más adelante), DISK-O-VIC se sitúa al principio de la memoria y quedará libre de interferencia del programa Basic. De esta forma, se añaden trece nuevos comandos sin afectar al sistema operativo normal.

La característica especial de carga también hace posible el uso de este paquete en un VIC-20 sea cual sea la cantidad de memoria adicional. No se vuelve anticuado aunque más adelante se añada una memoria adicional. Después de instalarse e inicializarse, la DISK-O-VIC consume 980 bytes y deja intacta la página cero.

Trece Comandos Nuevos en Disco

Antes de entrar en detalles sobre la manera de operación de la DISK-O-VIC, examinaremos los nuevos comandos para saber exactamente lo que hacen. (Para más detalles, vea la tabla de comandos).

Cada vez que un disco flexible se introduce en la 1541 y para luego tener acceso a dicho disco, se crea una tabla especial, llamada el mapa de disponibilidad de bloques, en la memoria de la unidad. Esta tabla contiene información especial sobre el disco que se encuentra en la unidad en un momento dado, como, por ejemplo, la forma en que se ha realizado la partición del disco, los bloques que se encuentran libres y otros asuntos de asignación. Afortunadamente, la unidad de discos se encarga de esta información un tanto esotérica, de forma que vd. se tendrá que preocupar pocas veces por ella.

El proceso de crear esta tabla se llama inicialización. Un disco tiene que inicializarse para que se escriba y se lea de forma correcta. (Tenga en cuenta que unos sistemas de unidad de discos que no son de Commodore utilizan el término inicialización para indicar "dar formato al disco", un proceso que puede escribir encima de

Listado 1. DISK-O-VIC, un programa de utilidad en disco para el VIC-20. Nota: Este programa utiliza el lenguaje ensamblador; se necesita un editor/ensamblador para ejecutarlo.

```
LINE# LOC   CODE        LINE

0000              ;********************************
0000              ;*                              *
0000              ;*          DISK-O-VIC          *
0000              ;*     DISK UTILITY PACKAGE     *.
0000              ;*   FOR THE VIC-20 COMPUTER    *
0000              ;*                              *
0000              ;*                              *
0000              ;*                              *
0000              ;*                              *
0000              ;*                              *
0000              ;*                              *
0000              ;********************************
0000              ;
0000              ;
0000              PARPTR = $0B        ,PUNTERO DE ANALIZADOR
0000              POINTR = $22        ,PUNTERO DE PROGRAMA DE UTILIDAD DEL VIC-20
0000              BASIC  = $2B        ,COMIENZO DE PUNTERO DE BASIC
0000              VARBLE = $2D        ,COMIENZO DE PUNTERO DE VARIABLES
0000              CHRGET = $73        ,RUTINA CHRGET EN BASIC
0000              CHRGOT = $79        ,"RE-GET" CARACTER ANTERIOR
0000              CHRPTR = $7A        ,PUNTERO CHRGET
0000              WEDGE  = $7E        ;BIFURCACION CHRGET DE BASIC
0000              ST     = $90        ,LINEA DE ESTADO "ST"
0000              CHRNOS = $B7        ,NUMERO DE CARACTERES EN NOMBRE
0000              DEVICE = $BA        ,DISPOSITIVO ACTUAL
0000              UTILPT = $FB        ,PUNTERO DE PROGRAMA DE UTILIDAD DISK-O-VIC
0000              FIRCHR = $FD        ,SALVAR PROVISIONALMENTE PARA CATALOGO
0000              SECCHR = $FE        ,SALVAR PROVISIONALMENTE PARA CATALOGO
0000              STACK  = $0100      ,"STACK" DEL SISTEMA
0000              BUFFER = $0200      ,ALMACENAMIENTO INTERMEDIO DE ENTRADA
0000              CMDBUF = $033C      ,ALMACENAM. INTERM. DE COMANDOS EN DISCO
0000              FLAG   = CMDBUF     ,DISTINTIVO PARA CATALOGO
0000              WARMST = $C474      ,REARRANQUE DEL BASIC
0000              WAIT   = $C48C      ,ESPERAR COMANDO DE SALIDA
0000              INFIN  = $C49F      ,TERMINAR INTRODUCIR LINEA
0000              CHAIN  = $C533      ,RECONSTRUIR SERIE DE LINEAS EN BASIC
0000              CLR    = $C659      ,AJUSTAR PUNTEROS, RESTAURAR BASIC
0000              INTEGR = $C96B      ,RECOGER ENTRADA DE ENTEROS
0000              PSTRNG = $CB1E      ,IMPRIMIR SERIE SEÑALADA POR (A,Y)
0000              ERROR  = $CF08      ,RUTINA DE ERROR DE SINTAXIS
0000              PRLINE = $DDCD      ,IMPRIMIR NUMERO DE LINEA
0000              CHROUT = $E742      ,REPRESENTAR EN PANTALLA
0000              RESET  = $FD22      ,ARRANQUE DEL VIC-20
0000              SECLIS = $FF93      ,SECUNDARIO DESPUES DE ESCUCHAR
0000              TKSA   = $FF96      ,SECUNDARIO DESPUES DE HABLAR
0000              ACPTR  = $FFA5      ,ACEPTAR DATOS DE RS-232
0000              CIOUT  = $FFA8      ,BYTE DE SALIDA A RS-232
0000              UNTALK = $FFAB      ,DESCONECTA EL PERIFERICO
0000              UNLIST = $FFAE      ,DESCONECTA EL PERIFERICO
0000              LISTEN = $FFB1      ,ORDENAR DISPOSITIVO QUE HABLE
0000              TALK   = $FFB4      ,ORDENAR DISPOSITIVO QUE HABLE
0000              SETLFS = $FFBA      ,FIJAR FICHERO LOGICO
0000              SETNAM = $FFBD      ,FIJAR EL NOMBRE DEL FICHERO
0000              OPEN   = $FFC0      ,ABRIR UN FICHERO
0000              CLOSE  = $FFC3      ,CERRAR UN FICHERO
0000              CLRCHN = $FFCC      ,RESTAURAR DISPOSITIVOS DE FALLOS
0000              INPUT  = $FFCF      ,INTRODUCIR UN BYTE
0000              OUTPUT = $FFD2      ,SALIDA DE BYTE
0000              LOAD   = $FFD5      ,CARGAR RAM DEL DISPOSITIVO
0000              SAVE   = $FFD8      ,SALVAR RAM EN DISPOSITIVO
0000              STOP   = $FFE1      ,COMPROBAR TECLA DE STOP
0000              GETIN  = $FFE4      ,OBTENER PULSACION
0000              ;
0000              ;
0000                     * = $1000
1000              ;
1000              ;
1000              ;INICIALIZACION DISK-O-VIC
1000              ;
1000              ;
1000  A2 00              LDX #SYSTM1-SYSTM1  ;INDICE NUEVO SISTEMA
1002  A0 00       CROSS   LDY #$00
1004  BD 54 13    PUTIN   LDA SYSTM1,X   ;CARGAR CODIGO OPERATIVO E
1007  F0 07               BEQ INSTAL     ;INSTALAR EN BIFURCACION
1009  99 7E 00            STA WEDGE,Y    ;CERO SIGNIFICA FIN DE BIFURCACION
100C  C8                  INY
100D  E8                  INX            ;INTRODUCIR SIGUIENTE CODIGO OPERATIVO
100E  D0 F4               BNE PUTIN      ;BIFURCAR SIEMPRE
1010  4C 74 C4    INSTAL  JMP WARMST     ;SALTAR A REARRANQUE
1013              ;
1013              ;
1013              ;COMIENZO DE INTRODUCCION DE BIFURCACION
1013              ;
1013              ;
1013  B0 0A       INSERT  BCS NORM       ;FIN DE CHRGET NORMAL
1015  C9 20               CMP #$20
1017  F0 32               BEQ REGET      ;IGNORAR ESPACIOS
1019  38                  SEC
101A  E9 30               SBC #$30
101C  38                  SEC
101D  E9 D0               SBC #$D0
101F  8D BE 13    NORM    STA ASAVE      ;SALVAR ACUMULADOR
1022  08                  PHP
1023  68                  PLA
```

Sigue →

datos o destruirlos. Esto no es el caso con la unidad de disco 1541).

Para asegurarse de que la información contenida en la memoria de la unidad esté actualizada, el disco se debe de inicializar con frecuencia durante una sesión. Según sale de la fábrica, la 1541 normalmente realiza el proceso de auto-inicialización durante la ejecución de varios comandos. Con la finalidad de tener un margen de seguridad, un proceso automático de inicialización precede cada mandato en DISK-O-VIC. Aunque esto parezca un poco excesivo, el proceso sólo tarda un segundo y ayuda enormemente a reducir el número de problemas. Nunca hace daño inicializarse de sobra.

Los comandos DLOAD y DSAVE funcionan exactamente igual que los comandos "Load" y "Save" del VIC-20, sólo que el ordenador sabe automáticamente que el dispositivo correcto para tener acceso es la unidad de disco (dispositivo número ocho). Estos comandos solamente se destinan a programas Basic.

Nunca se deben usar los comandos DLOAD o DSAVE en programas de

---

Tabla 1. Explicación de los comandos de DISK-O-VIC

"APPEND" (Agregar). Este comando permite que un programa que se encuentra en disco se agregue al final de otro en la memoria. Para que esto no se complique demasiado para el VIC-20, es importante que el programa en la memoria tenga los números de línea menores que los que tiene el programa en disco que se agrega. La disponibilidad de este comando facilita la construcción de grandes bibliotecas de subrutinas a partir de las cuales se pueden construir programas completos. La sintaxis correcta es:
APPEND "título de programa" (retorno)

Igual que ocurre con el Basic normal del VIC-20, se pueden abreviar algunas palabras. Por ejemplo, en vez de teclear la palabra completa APPEND, se puede teclear "A shift-P". (Todos los comandos DISK-O-VIC se pueden abreviar de esta forma. Simplemente se teclea la primera letra del comando, seguida por la segunda letra en "shift").

CATALOG (Catalogar). Para saber lo que contiene el disco que se encuentra actualmente en la unidad de disco, se mecanografía CATALOG y se pulsa la tecla de retorno. Un listado de directorio se imprime en la pantalla demostrando todos los programas disponibles. Hay que tener en cuenta que, a diferencia del método empleado normalmente por la 1541 para consultar el directorio, CATALOG no molesta el programa que se encuentra en la memoria del VIC-20.

Se ha agregado una característica especial de hacer una pausa, para que resulte más fácil. Se pulsa la barra de espaciado una vez para hacer una pausa en el listado. Se vuelve a pulsar para continuar. También se puede pulsar la tecla "run/stop" para parar un listado.

COLLECT (Recoger). Si se mecanografía este comando y se pulsa la tecla de retorno, el disco que se encuentra en la unidad será validado o recogido. En términos sencillos, esto hace que la unidad busque por todo el disco, comprobando que todos los bloques se han "conectado" correctamente. Cualquier bloque que ha sido asignado de forma incorrecta será despejado y preparado para servir de almacenamiento. La operación completa de este comando es bastante compleja, pero fundamentalmente lo que hace es revisar y despejar el disco. Igual que la operación de inicialización, nunca hace daño realizar el comando "Collect" con frecuencia.

DLOAD. Esto funciona igual que el comando normal de LOAD pero remite el fallo automáticamente a la unidad de discos. Por ejemplo, se mecanografía el "nombre del programa" DLOAD y se pulsa la tecla de retorno. La unidad se inicializa automáticamente, y se comprueban el programa cargado y los errores de disco. Para poner esto en perspectiva podemos decir que DLOAD es equivalente a los siguientes pasos:

OPEN 1, 8, 15, "I"
LOAD "program name", 8
INPUT#1 disk error message, etc.
CLOSE 1

Es evidente que aunque DLOAD es un comando simple, realiza mucho. Además, DLOAD sólo puede ser utilizado en programas Basic o programas de lenguaje de máquina que se "parecen" a Basic. Esta limitación se debe al hecho de que el VIC-20 tiene un extremo formado de carga de "memoria deslizante". (sliding memory)

DSAVE. Esto es igual que DLOAD pero salva el programa Basic en disco. Se realizan las mismas operaciones de inicialización y detección de errores.

HEADER. Esto es un comando especial que da formato a un disco virgen para su uso posterior. Se imprimen marcas magnéticas en el disco que sirven de guía al 1541, y se asignan un título y un código de identificación al disco. La sintaxis para su uso es la siguiente:

HEADER. "nombre del disco", Ixx (return)
donde "nombre del disco" es el nombre asignado al disco. "xx" se utiliza aquí como el código de identificación. Sin embargo, se puede utilizar cualquier combinación de dos caracteres. Tenga en cuenta que la coma es necesaria, igual que la letra "I". Antes de realizar un comando HEADER en el disco, se imprime la pregunta "Estás seguro? (Sí/No)" en la pantalla. Una respuesta de *"S" (SI) inicia el comando; cualquier otra* respuesta cancela anormalmente el proceso. Dado que el comando HEADER escribe encima del disco, es importante proporcionar esta característica de "Estas seguro?".

INIT. Como hemos indicado antes, cada comando de DISK-O-VIC incorpora la operación de inicialización automática. Sin embargo, hay ocasiones en que un disco está causando problemas y resulta necesario forzar un proceso de inicialización. Para hacer esto, se mecanografía INIT, se pulsa la tecla retorno y el disco será inicializado. Este comando es igual que teclear:
OPEN 1, 8, 15, "I"
CLOSE 1

KILL. Este es un comando de autodestrucción. Si vd. se ha hartado de utilizar DISK-O-VIC durante una sesión de programación y quiere eliminarlo totalmente del ordenador, teclee KILL y pulsa la tecla de retorno. El ordenador ejecutará un proceso entero de restauración, comportándose como si se hubiera apagado y vuelto a encenderse. No confunda este comando con el de "OFF" (a continuación). KILL restaura el ordenador totalmente. En general, utilice este comando solamente cuando quiere provocar un arranque.

OFF. Este comando apaga DISK-O-VIC, pero sigue dentro de la memoria, a salvo y protegido. Así puede volver a usarse en cualquier momento. Dado que DISK-O-VIC reduce un poco la velocidad de Basic, se puede apagar cuando se ejecuta un programa para poder alcanzar la máxima velocidad. Para volver a usarlo se teclea:
SYS 256*PEEK(56)+PEEK(55) (return)

RENAME. Este comando vuelve a nombrar un programa en disco, sin afectar ningún programa que se encuentre en la memoria. Por ejemplo,
RENAME "nombre antiguo" TO "nombre nuevo" (return)
cambiará el nombre del programa al "nombre nuevo". Hay que tener en cuenta varias cosas. El nombre antiguo viene primero, y luego el nuevo. La palabra "TO" tiene que estar presente entre los dos nombres para que funcione el comando. Finalmente, se proporciona una facilidad de detección de errores para que resulte imposible volver a nombrar un fichero con un nombre que se está utilizando.

SCRATCH. Con este comando se puede borrar (SCRATCH) un fichero o programa del disco. Se mecanografía SCRATCH seguido del nombre del fichero, y se pulsa la tecla de retorno. Una vez más, se presenta la pregunta "Estás seguro? (Y/N)". Se borra el fichero si se recibe la respuesta "Y".

SEND (Enviar). Este es un comando de fines generales y se puede utilizar para enviar los comandos en disco estandard de Commodore a la unidad. Por ejemplo,
SEND "I" (return)
enviará la letra I al disco, así provocando una inicialización. (Por supuesto, el comando INIT, de DISK-O-VIC, realiza la misma función). Otro ejemplo,
SEND "R: nombre nuevo = nombre antiguo" (return)
hará que se vuelva a nombrar el fichero "nombre antiguo". Dado que otros comandos en DISK-O-VIC sirven para casi todo, el comando SEND no se utiliza mucho. Pero resulta útil tenerlo a mano para las operaciones más avanzadas de programación en disco. Para que se sepa, Send es equivalente a
OPEN 1, 8, 15, "command"
CLOSE1

STATUS (Estado). Este comando busca la causa de un defecto en la operación de disco. Si se enciende la luz roja de detección de errores en la unidad de discos, se teclea STATUS y se pulsa la tecla de retorno. Se apaga la luz y un mensaje de error se imprime en la pantalla. El mensaje describe el error y donde se encuentra el problema en el disco (en términos de pista y sector). Si no hay ningún problema no se imprime ningún mensaje. Para comprobar este comando, teclee lo siguiente:
OPEN 1, 8, 1, "BASURA" (return)
La unidad 1541 empezará a funcionar, y suponiendo que no existe ningún fichero nombrado "BASURA" en el disco, la luz de errores se debe de encender. Teclee STATUS, y el mensaje de error se imprimirá en la pantalla. Vea el manual para la unidad de discos 1541 para consultar una explicación completa de los mensajes de error.

lenguaje de máquina o híbridos, dado que los comandos llegan a unas conclusiones sobre el comienzo de espacio del programa que pueden o no ser ciertas para programas de lenguaje de máquina. En general, todos los comandos de DISK-O-VIC dan por hecho que se trabaja en Basic.

Los comandos DLOAD y DSAVE automáticamente comprueban el canal de errores una vez realizada una operación para ver si todo ha salido bien. Si se detecta un error (Unidad No Preparada, Fichero Existe, Fichero No Encontrado, etc.), el mensaje se imprime en la pantalla y el fichero se cierra.

"Catálogo" constituye un comando interesante. Al contrario de cómo se hacían las cosas antes, se puede imprimir el directorio de discos o catálogo directamente en la pantalla, así conservando cualquier programa en la memoria. Para concluir la representación del listado en la pantalla, la barra de espaciado se pulsa una vez; para volver a iniciar la representación del listado, la barra de espaciado se vuelve a pulsar.

La finalidad del resto de los comandos debe ser evidente. En caso contrario sería útil repasar la Tabla 1 y consultar el manual de la unidad de discos 1541 de vez en cuando. Los usuarios de los ordenadores Commodore más grandes (y más caros) probablemente conocerán muchos de los comandos. A diferencia del VIC-20, ordenadores como el PET y SUPER-PET ya disponen de un juego de comandos un disco que se parece mucho a los de la DISK-O-VIC.

Cómo Funciona el Programa

Ahora que DISK-O-VIC se ha presentado, vamos a ver cómo funciona. Para que sea de más fácil comprensión, el Listado 1 presenta un listado de ensamblador para el programa completo. Dado que los ensambladores empiezan a ser más comunes para el VIC-20, a lo mejor vd. prefiere introducir el código fuente y ensamblar* su propia versión. Pero el listado de ensamblador se presenta aquí dado su valor didáctico, y la mayoría de los usuarios seguramente prefieren introducir el código objeto directamente. Se presenta un volcado hexadecimal de dicho código en el Listado 2.

Probablemente muchos problemas complicados se pueden resolver estudiando los comentarios en este listado. Sin embargo, para que sirva de ayuda en comprenderlo, describiré la estructura básica del programa. Para poder hacerlo, tenemos que estudiar la forma en que Basic recoge y ejecuta un comando.

Cuando actúa el Basic interpretativo, un puntero tiene que buscar los

Listado 1 continuación

```
1024  BD BD 13          STA SSAVE        ;SALVAR REGISTRO DE ESTADO
1027  BE BF 13          STX XSAVE        ;SALVAR REGISTRO X
102A  BC C0 13          STY YSAVE        ;SALVAR REGISTRO Y
102D  BA                TSX
102E  BD 01 01          LDA STACK+1,X    ;EXAMINAR DIRECCIONES DESPLAZADAS
1031  C9 8C             CMP #<WAIT       ;PARA VER SI LLEGAMOS DE
1033  D0 07             BNE COMMON       ;"ESPERAR" (ESPERANDO UNA
1035  BD 02 01          LDA STACK+2,X    ;RUTINA DE COMANDO)
1038  C9 C4             CMP #>WAIT
103A  F0 12             BEQ VCDISK       ;SI, LO HICIMOS, SIGUE
103C  AC C0 13   COMMON LDY YSAVE        ;NO, NO LO HICIMOS
103F  AE BF 13          LDX XSAVE        ;TIENE QUE SER UNA OPERACION
1042  AD BD 13          LDA SSAVE       ;NORMAL DE CHRGET, ASI QUE RESTAURAR TODOS
1045  48                PHA              ;LOS REGISTROS Y CONTINUAR
1046  AD BE 13          LDA ASAVE
1049  28                PLP
104A  60                RTS
104B  4C 73 00   REGET  JMP CHRGET
104E             ;
104E             ;
104E             ;PARSER ROUTINE
104E             ;
104E             ;
104E  AD BD 13   VCDISK LDA SSAVE        ;MIRA EL ANTERIOR REGISTRO DE ESTADO
1051  4A                LSR A            ;ERA LA ENTRADA UNA LINEA'?
1052  90 3A             BCC NUMBER       ;SI. BUSCA EN OTRO SITIO
1054  A2 00             LDX #$00         ;NO, ERA UN COMANDO
1056  86 0B             STX PARPTR       ;INTRODUCIR POR INDICE EN LOS COMANDOS
1058  A4 7A      PARSER LDY CHRPTR       ;DIRECCION DE CARACTER EN COMANDO
105A  B9 00 02   CHECK  LDA BUFFER,Y     ;OBTENER EL CARACTER DE COMANDO
105D  38                SEC
105E  FD 5C 13          SBC KEYWRD,X     ;COMPARAR CON TABLA
1061  F0 13             BEQ DIRECT       ;TOTALMENTE DE ACUERDO
1063  C9 80             CMP #$80
1065  F0 13             BEQ SHIFT        ;COMPARAR LETRAS MEDIANTE "SHIFT"
1067  E6 0B             INC PARPTR       ;SEGUIR PISTA DE POSICION
1069  E8         MOVE   INX              ;NO LA PALABRA CLAVE NECESARIA, ASI QUE
106A  BD 5B 13          LDA KEYWRD-1,X ;DESPLAZAR X A UN PUNTO (PARA APUNTAR A
106D  10 FA             BPL MOVE         ;SIGUIENTE PALABRA DE LA TABLA
106F  BD 5C 13          LDA KEYWRD,X
1072  D0 E4             BNE PARSER       ;BUSCAR SIGUIENTE PALABRA CLAVE
1074  F0 C6             BEQ COMMON       ;PALABRAS CLAVE AGOTADAS
1076  E8         DIRECT INX              ;HASTA AHORA SON IGUALES, PERO
1077  C8                INY              ;COMPROBAR SIGUIENTE LETRA
1078  D0 E0             BNE CHECK        ;BIFURCAR SIEMPRE
107A  84 7A      SHIFT  STY CHRPTR       ;PUNTAR CHRPTR EN ULTIMO BYTE
107C  A5 0B             LDA PARPTR       ;BUSCAR POSICION DE COMANDO
107E  0A                ASL A            ;MULTIPLICAR POR DOS DADO QUE
107F  AA                TAX              ;DIRECCION SE COMPONE DE DOS BYTES
1080  BD A4 13          LDA ACTION+1,X ;CARGAR LA DIRECCION CORRECTA
1083  48                PHA              ;DE RUTINA DESPLAZANDOLA
1084  BD A3 13          LDA ACTION,X     ;EN EL "STACK"
1087  48                PHA
1088  20 3C 10          JSR COMMON       ;RESTAURAR TODOS LOS REGISTROS
108B  4C 73 00          JMP CHRGET       ;TERMINAR ENTRADA DE LINEAS
108E             ;
108E             ;
108E                    ;AQUI LA RUTINA SALTA A "CHRGET"
108E                    ;QUE SENALA "CHRPTE" EN SIGUIENTE BYTE
108E                    ;AL FINAL SE ENCONTRARA LA SUBRUTINA LLAMADA
108E                    ;"NORMAL". CUANDO SE PULSA "RTS"
108E                    ;AL FINAL, EL CONTROL SE TRANSFIERE A LA
108E                    ;DIRECCION DE ACCION DE COMANDOS DESPLAZADA EN EL "STACK"
108E             ;
108E             ;
108E  68         NUMBER PLA                  ;INCAPACITAR DIRECCION NORMAL RTS
108F  68                PLA
1090  20 3C 10          JSR COMMON           ;RESTAURAR TODOS LOS REGISTROS
1093  20 6B C9          JSR INTEGR           ;OBTENER NUMERO DE LINEA
1096  4C 9F C4          JMP INFIN            ;TERMINAR ENTRADA DE LINEA
1099             ;
1099             ;
1099             ;*** ;"APPEND", ENTRADA DE COMANDO ***
1099             ;
1099             ;
1099  A0 02      APPEND LDY #$02
109B  A5 2D      DECREM LDA VARBLE           ;REDUCIR DOS VECES EL FINAL
109D  D0 02             BNE ONCE             ;DEL PUNTERO DE PROGRAMA
109F  C6 2E             DEC VARBLE+1
10A1  C6 2D      ONCE   DEC VARBLE
10A3  88                DEY
10A4  D0 F5             BNE DECREM           ;VOLVER A HACERLO SI HACE FALTA
10A6  A5 2D             LDA VARBLE           ;A CONTINUACION SEÑALA LA RUTINA DE
10A8  85 FB             STA UTILPT           ;"APPEND" A ESTA POSICION
10AA  A5 2E             LDA VARBLE+1
10AC  85 FC             STA UTILPT+1
10AE  4C 3C 11          JMP LOADSB           ;REALIZAR CAMBIO DE POSICION DE CARGA
10B1             ;
10B1             ;
10B1             ;*** ;COMANDO DE ENTRADA "CATALOGO" ***
10B1             ;
10B1             ;
10B1  20 EA 12   CAT    JSR ICMD             ;INICIALIZAR DISCO
10B4  A9 24             LDA #'$              ;COMANDO PARA CATALOGO
10B6  8D 3C 03          STA CMDBUF
10B9  A2 3C             LDX #<CMDBUF         ;OBTENER DIRECCION DE COMANDO
10BB  A0 03             LDY #>CMDBUF
10BD  A9 01             LDA #$01             ;LONGITUD DE COMANDO
10BF  20 BD FF          JSR SETNAM           ;FIJAR EL NOMBRE
10C2  A9 0E             LDA #$0E             ;NUMERO DE FICHERO
```

Sigue

Listado 1 continuación

```
00196  10C4  A0 60              LDY #$60        ;DIRECCION SECUNDARIA
00197  10C6  A2 08              LDX #$08        ;NUMERO DE DISPOSITIVO
00198  10C8  20 BA FF           JSR SETLFS      ;FIJAR EL FICHERO LOGICO
00199  10CB  20 C0 FF           JSR OPEN        ;ABRIR EL FICHERO
00200  10CE  A9 08              LDA #$08
00201  10D0  20 B4 FF           JSR TALK        ;INICIAR CONVERSACION
00202  10D3  A9 60              LDA #$60        ;ENVIAR EL COMANDO AHORA
00203  10D5  20 96 FF           JSR TKSA
00204  10D8  A9 00              LDA #$00        ;SACAR EL ST EN CERO
00205  10DA  85 90              STA ST
00206  10DC  A0 03              LDY #$03        ;SALTAR LAS DIRECCIONES INICIALES
00207  10DE  8C 3C 03    THREAD  STY FLAG        ;SALVAR EL CONTADOR DE SALTOS
00208  10E1  20 A5 FF           JSR ACPTR       ;OBTENER UN CARACTER
00209  10E4  85 FD              STA FIRCHR      ;SALVAR PRIMER CARACTER
00210  10E6  A4 90              LDY ST          ;COMPROBAR EL ST
00211  10E8  D0 39              BNE BADSTA      ;BIFURCAR SI VAN MAL
00212  10EA  20 A5 FF           JSR ACPTR       ;OBTENER SIGUIENTE CARACTER
00213  10ED  85 FE              STA SECCHR      ;SALVAR SEGUNDO CARACTER
00214  10EF  A4 90              LDY ST          ;VOLVER A COMPROBAR
00215  10F1  D0 30              BNE BADSTA      ;BIFURCAR SI VA MAL
00216  10F3  AC 3C 03           LDY FLAG        ;CUANTOS BYTES
00217  10F6  88                 DEY             ;HEMOS RECHAZADO?
00218  10F7  D0 E5              BNE THREAD      ;NO LOS SUFICIENTES
00219  10F9  A6 FD              LDX FIRCHR      ;OBTENER PRIMER CARACTER
00220  10FB  A5 FE              LDA SECCHR      ;OBTENER SEGUNDO CARACTER
00221  10FD  20 CD DD           JSR PRLINE      ;IMPRIMIR EL NUMERO DE BLOQUES
00222  1100  20 39 13           JSR SPACE       ;IMPRIMIR UN ESPACIO
00223  1103  20 A5 FF   CONTEN  JSR ACPTR       ;OBTENER CARACTER
00224  1106  A6 90              LDX ST          ;COMPROBAR EL ST
00225  1108  D0 19              BNE BADSTA      ;SEGUIR ADELANTE SI SE HA HECHO
00226  110A  C9 00              CMP #$00        ;O=FIN DE LINEA
00227  110C  F0 0E              BEQ FINLIN      ;TERMINAR LA LINEA
00228  110E  20 D2 FF           JSR OUTPUT      ;SI NO, IMPRIMIR CARACTER
00229  1111  20 E1 FF           JSR STOP        ;COMPROBAR LA TECLA STOP
00230  1114  F0 0D              BEQ BADSTA      ;BIFURCAR SI SE EMPUJA
00231  1116  20 27 13           JSR PAUSE       ;PAUSAR SI HACE FALTA
00232  1119  4C 03 11           JMP CONTEN      ;IMPRIMIR SIGUIENTE ENTRADA
00233  111C  20 36 13   FINLIN  JSR CARRET      ;IMPRIMIR RETORNO DE CARRO
00234  111F  A0 02              LDY #$02        ;SEGUIR A LINEA SIGUIENTE
00235  1121  D0 BB              BNE THREAD      ;BIFURCAR SIEMPRE
00236  1123  A9 0E      BADSTA  LDA #$0E        ;CERRAR FICHERO
00237  1125  20 C3 FF           JSR CLOSE
00238  1128  20 CC FF           JSR CLRCHN      ;RESTAURAR DISPOSITIVOS DE FALLOS
00239  112B  4C 74 C4           JMP WARMST      ;RETORNAR A BASIC
00240  112E                     ;
00241  112E                     ;
00242  112E                     ;*** ;ENTRADA DE COMANDO "RECOGER". ***
00243  112E                     ;
00244  112E                     ;
00245  112E  20 ED 12   COLLEC  JSR CCMD        ;REALIZAR RECOGIDA
00246  1131  4C 74 C4           JMP WARMST      ;RETORNAR A BASIC
00247  1134                     ;
00248  1134                     ;
00249  1134                     ;*** ;ENTRADA DE COMANDO "DLOAD" ***
00250  1134                     ;
00251  1134                     ;
00252  1134  A5 2B      DLOAD   LDA BASIC       ;EL PROGRAMA SE CARGARA
00253  1136  85 FB              STA UTILPT      ;EN AREA NORMAL
00254  1138  A5 2C              LDA BASIC+1     ;DE PROGRAMA BASIC
00255  113A  85 FC              STA UTILPT+1
00256  113C  20 AE 12   LOADSB  JSR PARAMS      ;OBTENER NOMBRE DE PROGRAMA
00257  113F  20 99 12           JSR SETDIS      ;FIJAR TODOS PARAMETROS DE DISCO
00258  1142  A9 00              LDA #$00        ;CERO SIGNIFICA CARGAR
00259  1144  A6 FB              LDX UTILPT      ;CAMBIAR POSICION DE CARGA
00260  1146  A4 FC              LDY UTILPT+1    ;A ESTE PUNTERO
00261  1148  20 D5 FF           JSR LOAD        ;REALIZAR LA CARGA
00262  114B  20 33 C5           JSR CHAIN       ;VOLVER A CONSTRUIR LA SERIE
00263  114E  A5 22              LDA POINTR      ;AJUSTAR COMIENZO DE VARIABLES
00264  1150  A6 23              LDX POINTR+1    ;PARA QUE APUNTE JUSTO DESPUES DE
00265  1152  18                 CLC             ;LOS TRES BYTES DE CERO AL
00266  1153  69 02              ADC #$02        ;FINAL DEL PROGRAMA
00267  1155  85 2D              STA VARBLE
00268  1157  90 01              BCC NOADJ
00269  1159  E8                 INX
00270  115A  86 2E      NOADJ   STX VARBLE+1
00271  115C  20 59 C6           JSR CLR         ;DESPEJAR TODOS LOS PUNTEROS, ETC
00272  115F  4C 79 11           JMP CLEAN       ;IR A DESPEJAR AL FICHERO
00273  1162                     ;
00274  1162                     ;
00275  1162                     ;*** ;ENTRADA DE MANDATO "DSAVE" ***
00276  1162                     ;
00277  1162                     ;
00278  1162  20 AE 12   DSAVE   JSR PARAMS      ;OBTENER NOMBRE DE PROGRAMA
00279  1165  20 99 12           JSR SETDIS      ;FIJAR PARAMETROS DE DISCOS
00280  1168  A5 2B              LDA BASIC       ;SEÑALAR "UTILPT" AL
00281  116A  85 FB              STA UTILPT      ;PRINCIPIO DE PROGRAMA
00282  116C  A5 2C              LDA BASIC+1
00283  116E  85 FC              STA UTILPT+1
00284  1170  A9 FB              LDA #<UTILPT    ;PAGINA EN DIRECCION DE "UTILPT"
00285  1172  A6 2D              LDX VARBLE      ;REALIZAR 0-X E Y CONTIENEN FIN
00286  1174  A4 2E              LDY VARBLE+1    ;DE DIRECCION DE PROGRAMA
00287  1176  20 D8 FF           JSR SAVE        ;SALVAR EN DISCO
00288  1179  20 E5 12   CLEAN   JSR SHUT        ;CERRAR EL FICHERO
00289  117C  20 53 12           JSR DERROR      ;COMPROBAR ERRORES EN DISCO
00290  117F  4C 74 C4           JMP WARMST      ;TODO HECHO
00291  1182                     ;
00292  1182                     ;
00293  1182                     ;*** ;ENTRADA DE COMANDO "HEADER" ***
00294  1182                     ;
00295  1182                     ;
00296  1182  20 FA 12   HEADER  JSR HDISK       ;PREPARAR COMANDO
```

Sigue

comandos analizando o barriendo la línea de entrada. El intérprete revisa la línea de entrada, carácter por carácter, en búsqueda de un comando que reconoce. Por lo tanto, para añadir comandos nuevos al Basic hay que colocar una bifurcación en la rutina del análisis, desviando la atención del procedimiento normal del barrido a un procedimiento nuevo. Realmente lo que ocurre es que el analizador está obligado a buscar primero los comandos nuevos. Si no puede comparar un comando con ninguno de la lista nueva, se vuelve a enviar el control al sistema normal donde el comando de entrada se comprueba contra la lista antigua.

El primer bloque de código del Listado 1, las líneas 00072-00080, es la rutina de inicialización. Este código introduce bifurcación en la rutina normal de análisis, y de esta forma el proceso de inicialización sólo tiene que tener lugar al principio de la sesión. Después del proceso de inicialización, el analizador Basic siempre busca primero los comandos.

DISK-O-VIC

El siguiente bloque de código se sitúa en las líneas 00086-00165. Esto es un suplemento del analizador. Como hemos indicado antes, el analizador se dirige a esta rutina cada vez que un comando entra en el ordenador. La instrucción clave en este bloque se encuentra en la línea 00100. El "stack" se analiza por si incluye alguna dirección de "Return" ("RTS"). Si la dirección al principio del "stack" indica que el analizador procede del estado de "esperando un mandato" del VIC-20, sí inicia la acción. Si se encuentra alguna otra dirección, el analizador puede continuar con sus actividades normales.

Suponiendo que la prueba se ha concluido con éxito y el VIC-20 realmente espera un comando, la línea de entrada se verifica carácter por carácter. Esto ocurre en el bloque de código etiquetado "Rutina de Análisis", que se encuentra en las líneas 00119-00150. La entrada se comprueba contra la lista de comandos DISK-O-VIC localizada en una tabla en las líneas 00573-00586. Si se encuentran dos iguales, se forma una "dirección de acción", y el control pasa a la subrutina correspondiente.

Subrutinas

La mayor parte del programa se destina a las numerosas subrutinas de comandos. Para que sean más fáciles de encontrar, estas subrutinas se clasifican en orden alfabético, empezando por "Append", luego "Catálogo", etc. Aunque en este momento el programa parezca complejo, la verdad es que resulta bastante fácil de analizar

Listado 1 continuación

```
00297  1185  20 B0 12         JSR STRING       ;AÑADIR NOMBRE DE DISCO.
00298  1188  20 73 00         JSR CHRGET       ;OBTENER SIGUIENTE CARACTER.
00299  118B  C9 2C            CMP #',          ;REALIZAR UN "NEED" DE UNA COMA.
00300  118D  D0 2D            BNE BAD1         ;ERROR DE SINTAXIS.
00301  118F  9D 3C 03         STA CMDBUF,X     ;INTRODUCIR LA COMA.
00302  1192  E8               INX
00303  1193  20 73 00         JSR CHRGET       ;OBTENER INDENTIDAD DE DISCO.
00304  1196  C9 49            CMP #'I          ;TIENE QUE TENER "I".
00305  1198  D0 22            BNE BAD1
00306  119A  20 73 00         JSR CHRGET       ;OBTENER NUMERO DE IDENTIDAD.
00307  119D  F0 1D            BEQ BAD1         ;MAL, SI NO SE ENCUENTRA ALLI.
00308  119F  9D 3C 03         STA CMDBUF,X     ;SALVAR PRIMER DIGITO.
00309  11A2  E8               INX
00310  11A3  20 73 00         JSR CHRGET       ;RECOGER SIGUIENTE DIGITO.
00311  11A6  F0 14            BEQ BAD1         ;MAL, SI NO SE ENCUENTRA ALLI.
00312  11A8  9D 3C 03         STA CMDBUF,X     ;SALVAR SEGUNDO DIGITO.
00313  11AB  E8               INX
00314  11AC  20 73 00         JSR CHRGET       ;CARACTERES EXTRAÑOS?
00315  11AF  D0 0B            BNE BAD1         ;SI, MALAS NOTICIAS.
00316  11B1  9D 3C 03         STA CMDBUF,X     ;CERO SIGNIFICA FIN.
00317  11B4  20 0F 13         JSR SURE         ;ESTAS SEGURO?
00318  11B7  D0 09            BNE NOGO         ;NINGUN CERO SIGNIFICA NO.
00319  11B9  4C 32 12         JMP XMIT         ;SI, REALIZA HEADER.
00320  11BC  4C 08 CF  BAD1   JMP ERROR
00321  11BF                   ;
00322  11BF                   ;
00323  11BF                   ;*** ENTRADA DE COMANDO "INICIALIZAR" ***
00324  11BF                   ;
00325  11BF                   ;
00326  11BF  20 EA 12  INIT   JSR ICMD         ;REALIZAR INICIALIZACION.
00327  11C2  4C 74 C4  NOGO   JMP WARMST       ;TODO HECHO.
00328  11C5                   ;
00329  11C5                   ;
00330  11C5                   ;*** ENTRADA DE COMANDO "KILL" ***
00331  11C5                   ;
00332  11C5                   ;
00333  11C5  20 22 FD  KILL   JSR RESET        ;REALIZAR ARRANQUE.
00334  11C8                   ;
00335  11C8                   ;
00336  11C8                   ;*** 'OFF' ENTRADA DE COMANDO "OFF" ***
00337  11C8                   ;
00338  11C8                   ;
00339  11C8  A2 04     OFF    LDX #SYSTM2-SYSTM1 ;BORRAR ANALIZADOR ACTUAL.
00340  11CA  4C 02 10         JMP CROSS        ;COLOCAR CHRGET ANTERIOR EN SU SITIO
00341  11CD                   ;
00342  11CD                   ;
00343  11CD                   ;*** ENTRADA DE COMANDO "RENAME" ***
00344  11CD                   ;
00345  11CD                   ;
00346  11CD  20 FD 12  RENAME JSR RDISK        ;PREPARAR COMANDO
00347  11D0  20 79 00         JSR CHRGOT       ;OBTENER SIGUIENTE CARACTER.
00348  11D3  F0 E7            BEQ BAD1         ;NO SE DA NINGUN NOMBRE, MALO.
00349  11D5  C9 22            CMP #'"
00350  11D7  D0 E3            BNE BAD1         ;NINGUNA CITA DE COMIENZO ES MALO.
00351  11D9  A5 7A            LDA CHRPTR       ;SALVAR PUNTERO EN ANTIGUO NOMBRE.
00352  11DB  85 FB            STA UTILPT
00353  11DD  A5 7B            LDA CHRPTR+1
00354  11DF  85 FC            STA UTILPT+1
00355  11E1  20 73 00  TWIST  JSR CHRGET       ;PASAR POR NUEVO NOMBRE.
00356  11E4  F0 D6            BEQ BAD1         ;MALO SI NO SE DA NINGUNO.
00357  11E6  C9 22            CMP #'"          ;BUSCAR FINAL DE CITA.
00358  11E8  D0 F7            BNE TWIST
00359  11EA  20 73 00         JSR CHRGET       ;A CONTINUACION BUSCAR PALABRA "TO"
00360  11ED  C9 54            CMP #'T
00361  11EF  D0 CB            BNE BAD1
00362  11F1  20 73 00         JSR CHRGET
00363  11F4  C9 4F            CMP #'O
00364  11F6  D0 C4            BNE BAD1
00365  11F8  20 73 00         JSR CHRGET       ;ENCONTRADO.
00366  11FB  20 B0 12         JSR STRING       ;COLOCAR PRIMERO EL ULTIMO NOMBRE
00367  11FE  A9 3D            LDA #'=          ;INTRODUCIR SEÑAL DE IGUAL A ( )
00368  1200  9D 3C 03         STA CMDBUF,X
00369  1203  E8               INX
00370  1204  A5 FB            LDA UTILPT       ;RESTAURAR CHRPTR ANTERIOR.
00371  1206  85 7A            STA CHRPTR
00372  1208  A5 FC            LDA UTILPT+1
00373  120A  85 7B            STA CHRPTR+1
00374  120C  20 B0 12         JSR STRING       ;COLOCAR PRIMER NOMBRE EL ULTIMO.
00375  120F  A9 00            LDA #$00         ;BYTE DE CERO PARA EL FINAL.
00376  1211  9D 3C 03         STA CMDBUF,X
00377  1214  4C 32 12         JMP XMIT         ;ENVIAR EL COMANDO.
00378  1217                   ;
00379  1217                   ;
00380  1217                   ;*** ENTRADA DE COMANDO "SCRATCH"
00381  1217                   ;
00382  1217                   ;
00383  1217  20 00 13  SCRATC JSR SDISK        ;FIJAR COMANDO "S".
00384  121A  20 B0 12         JSR STRING       ;AÑADIR NOMBRE.
00385  121D  A9 00            LDA #$00         ;COLOCAR BYTE DE CERO AL FINAL.
00386  121F  9D 3C 03         STA CMDBUF,X
00387  1222  20 0F 13         JSR SURE         ;ESTAS SEGURO?
00388  1225  D0 29            BNE EGRESS       ;NO, SI NO ES CERO.
00389  1227  4C 32 12         JMP XMIT         ;REALIZAR EL "SCRATCH".
00390  122A                   ;
00391  122A                   ;
00392  122A                   ;*** ENTRADA DE COMANDO "SEND" ***
00393  122A                   ;
00394  122A                   ;
00395  122A  20 AE 12  SEND   JSR PARAMS       ;OBTENER "STRING" DE COMANDOS
```

Sigue →

atacando una pequeña función a la vez.

Se presentan unas subrutinas de fines generales hacia el final de programa, en la línea 00452. Normalmente son utilizadas por el resto del programa para recoger los nombres de ficheros, obtener los parámetros de los discos, imprimir mensajes en la pantalla, etc. En general, se les han asignado etiquetas o nombres relacionados a las funciones que realizan.

DISK-O-VIC termina presentando varias tablas de datos y direcciones. Primero se presenta una tabla de palabras clave, como hemos indicado antes. A continuación se presenta una tabla que contiene las direcciones de las subrutinas de comandos. Al final, se crea un juego de variables con la finalidad de salvar registros, etc. Al asignar variables a esta zona, se evita el uso de las importantes posiciones de página cero.

Dado que los detalles a veces confunden el asunto, a continuación se presenta un resumen de la estructura general que acabamos de describir:

—Proceso de inicialización.

—La bifurcación se introduce en el analizador.

—La nueva rutina de análisis.

—Las subrutinas de comandos.

—Las subrutinas de fines generales.

—Los datos, direcciones y variables.

Antes de dejar los aspectos teóricos de DISK-O-VIC, debemos mirar la tabla de ecuaciones en el listado de ensamblador. La tabla utiliza alrededor de una docena de posiciones de página cero, pero dado que se utilizan para sus fines normales, el sistema operativo les hace poco caso. La verdad es que las posiciones $FB a $FE son de página cero y el VIC-20 no las utiliza en absoluto.

Dado que se trata de un sistema operativo en disco, se supone que la unidad de cassettes no se utiliza. Esto libera un gran bloque de espacio, empezando por $033C, que normalmente se emplea como almacenamiento intermedio de cassette. Por lo tanto este bloque se puede utilizar como almacenamiento intermedio para los comandos en disco. Incluso si una unidad de cassettes se une al VIC-20 junto con un disco flexible, no surgirá ningún conflicto dado que pocas veces se tendría acceso a las dos unidades a la vez.

Rutinas ROM

La siguiente en las tablas de ecuaciones constituye un grupo grande de subrutinas incluidas en el conjunto ROM del sistema operativo VIC-20. Aunque vd. no tenga un uso inmediato para el DISK-O-VIC, esta tabla le será útil. El uso de una de estas rutinas ROM estandard en programas

fabricados por uno mismo, permitirá el ahorro de centenares, hasta miles, de bytes. Vamos a examinar unas cuantas rutinas con más detalles.

Las rutinas ROM del VIC-20 pueden clasificarse en dos categorías generales. La primera, llamada Kernal, incluye las rutinas principales para operaciones de entrada y salida. (Aunque no se sepa por qué, la forma de escribir "Kernal" es la que se acepta oficialmente en Commodore). Estas rutinas Kernal se consideran especiales porque varios modelos de ordenadores Commodore incluyen las mismas rutinas que se encuentran en las mismas direcciones.

Por ejemplo, la rutina de "salida de byte" tiene lugar en $FFD2 en todos los modelos de ordenadores Commodore. Sin embargo, en general las rutinas Kernal son idénticas solamente en el VIC-20 y el Commodore 64. Esto significa que el software que emplea mucho las rutinas Kernal debe ser de fácil transferencia del VIC-20 al Commodore 64, y al revés.

Los comandos Basic forman otra categoría de rutinas. Las posiciones de éstos serán diferentes en cada máquina, pero parecidos para los ordenadores Commodore. Un ejemplo de estas rutinas es la que se encuentra en $CB1E, etiquetada PSTRNG en el listado 1. Cuando se solicita, se imprime un "string" en pantalla señalado por el acumulador y el registro "y". Sigue imprimiendo la serie de caracteres hasta que se detecta un cero final.

La tabla de ecuaciones del listado 1 proporciona información adicional sobre el funcionamiento del VIC-20. En la mayoría de los casos, las posiciones y las rutinas llevan etiquetas para facilitar la interpretación, y los comentarios proporcionan más detalles. Si desea saber más sobre las rutinas Kernal, vea "La Guía de Referencia para el Programador del VIC-20 (Howard W. Sams and Co., PO Box 7092, Indianapolis, IN 46206). Para poder comprender las rutinas no Kernal, hace falta un mapa de memoria más extensa.

Introducir el Programa

Ya que hemos hablado de la teoría y la operación del DISK-O-VIC, tenemos que considerar el lado práctico de las cosas. El usuario debe de crear una copia en disco del código objeto para que tenga el programa utilidad siempre a mano. Con esta finalidad, el volcado hexadecimal del Listado 2 corresponde al código fuente del Listado 1. Para usarlo, se introducen los números hexadecimales en el VIC-20 y se salva en disco. Por lo tanto, para llamar DISK-O-VIC, se carga y se inicializa el código.

Listado 1 continuación

```
122D  A9 00            LDA #$00         ;BYTE DE CERO SIGNIFICA
122F  9D 3C 03         STA CMDBUF,X     ;FIN DE "STRING"
1232  20 EA 12  XMIT   JSR ICMD         ;INICIALIZAR DISCO.
1235  20 DB 12         JSR RESPON       ;FIJAR CANAL DE COMANDOS.
1238  A0 00            LDY #$00
123A  B9 3C 03  SHOOT  LDA CMDBUF,Y     ;OBTENER "STRING" DE COMANDOS
123D  F0 06            BEQ SWEEP        ;CERO SIGNIFICA FINAL.
123F  20 A8 FF         JSR CIOUT        ;SI NO, ENVIAR CARACTER.
1242  C8               INY
1243  90 F5            BCC SHOOT        ;BIFURCAR SI NO EXISTE ERROR.
1245  20 AE FF  SWEEP  JSR UNLIST       ;DESCONECTAR EL DISCO.
1248  20 53 12         JSR DERROR       ;COMPROBAR ESTADO DE DISCO.
124B  F0 03            BEQ EGRESS       ;BIFURCAR SIEMPRE.
124D                   ;
124D                   ;
124D                   ;*** ;ENTRADA DE COMANDO "STATUS" ***
124D                   ;
124D                   ;
124D  20 53 12  STATUS JSR DERROR       ;COMPROBAR ESTADO DISCO.
1250  4C 74 C4  EGRESS JMP WARMST
1253  A9 08     DERROR LDA #$08         ;PREPARADO PARA LEER
1255  85 BA            STA DEVICE       ;EL CANAL DE ERRORES.
1257  20 B4 FF         JSR TALK         ;OBLIGAR EL BUS QUE HABLE.
125A  A9 6F            LDA #$6F         ;ESTO ES CANAL DE ERRORES.
125C  20 96 FF         JSR TKSA         ;DIRECCION SECUNDARIA DESPUES DE HABLAR.
125F  A0 00            LDY #$00
1261  20 A5 FF  XFER   JSR ACPTR        ;OBTENER BYTE DE BUS.
1264  99 3C 03         STA CMDBUF,Y     ;SALVARLO AQUI.
1267  C8               INY
1268  C9 0D            CMP #$0D         ;BUSCAR RETORNO DEL CARRO.
126A  D0 F5            BNE XFER         ;SI NO, OBTENER SIGUIENTE CARRO.
126C  A9 00            LDA #$00         ;COLOCAR BYTE DE CERO PARA EL FINAL.
126E  99 3C 03         STA CMDBUF,Y
1271  20 AB FF         JSR UNTALK       ;DESCONECTAR CANAL.
1274  A0 00            LDY #$00         ;BUSCAR "OO" DE ASCII
1276  A9 30            LDA #'O
1278  D9 3C 03         CMP CMDBUF,Y
127B  D0 06            BNE DEFER        ;SIN DUDA ES UN ERROR.
127D  C8               INY
127E  D9 3C 03         CMP CMDBUF,Y
1281  F0 15            BEQ NOERR        ;NO SE ENCUENTRA NINGUN ERROR.
1283  20 E5 12  DEFER  JSR SHUT         ;CERRAR EL FICHERO.
1286  20 CC FF         JSR CLRCHN       ;RESTAURAR DISPOSITIVOS DE FALLOS.
1289  20 36 13         JSR CARRET       ;IMPRIMIR RETORNO DEL CARRO.
128C  A9 3C            LDA #<CMDBUF
128E  A0 03            LDY #>CMDBUF
1290  20 1E CB         JSR PSTRNG       ;IMPRIMIR EL MENSAJE DE ERROR.
1293  68               PLA              ;NO RETORNAR SI ES MALO.
1294  68               PLA
1295  4C 74 C4         JMP WARMST       ;IR A BASIC.
1298  60        NOERR  RTS              ;RETORNAR A RUTINA DE LLAMADA.
1299                   ;
1299                   ;
1299                   ; SUBRUTINAS DE FINES GENERALES.
1299                   ;
1299                   ;
1299  20 EA 12  SETDIS JSR ICMD         ;INICIALIZAR DISCO.
129C  BA               TXA              ;ESTO ES LA LONGITUD DE NOMBRE.
129D  A2 3C            LDX #<CMDBUF     ;X E Y CONTIENEN
129F  A0 03            LDY #>CMDBUF     ;LA DIRECCION DEL NOMBRE.
12A1  20 BD FF         JSR SETNAM       ;FIJAR NOMBRE DE PROGRAMA.
12A4  A9 08            LDA #$08         ;NUMERO DE FICHERO LOGICO.
12A6  AA               TAX              ;NUMERO DE DISPOSITIVO DE DISCO.
12A7  A0 00            LDY #$00         ;SECUNDARIO PARA VOLVER A UBICAR.
12A9  84 90            STY ST           ;SACAR EN CERO "ST".
12AB  4C BA FF         JMP SETLFS       ;FIJAR PARAMETROS DE FICHERO.
12AE                   ;
12AE                   ;
12AE  A2 00     PARAMS LDX #$00
12B0  86 B7     STRING STX CHRNOS       ;SALVAR LONGITUD ORIGINAL.
12B2  20 79 00         JSR CHRGOT       ;OBTENER NOMBRE DE FICHERO.
12B5  F0 1F            BEQ BAD2         ;NO SE DA NINGUN NOMBRE.
12B7  C9 22            CMP #'"          ;SE NECESITA UNA CITA.
12B9  D0 1B            BNE BAD2
12BB  E6 7A     PICKUP INC CHRPTR       ;ACTUALIZAR PUNTERO CHRGET.
12BD  D0 02            BNE NOZER
12BF  E6 7B            INC CHRPTR+1
12C1  A0 00     NOZER  LDY #$00
12C3  B1 7A            LDA (CHRPTR),Y   ;OBTENER SIGUIENTE CARACTER.
12C5  F0 0A            BEQ DONE         ;CERO SIGNIFICA EL FINAL DEL ALMAC. INTERM.
12C7  C9 22            CMP #'"          ;CITA FINAL AHORA?
12C9  F0 06            BEQ DONE         ;SI, "STRING" OBTENIDO.
12CB  9D 3C 03         STA CMDBUF,X     ;ALMACENAR AQUI PROVISIONALMENTE.
12CE  E8               INX
12CF  D0 EA            BNE PICKUP       ;BIFURCAR SIEMPRE.
12D1  E4 B7     DONE   CPX CHRNOS       ;NO PERMITIR NOMBRES DE
12D3  F0 01            BEQ BAD2         ;LONGITUD CERO.
12D5  60               RTS
12D6  68        BAD2   PLA              ;QUITAR ANTIGUA DIRECCION DE RETORNO.
12D7  68               PLA
12D8  4C 08 CF         JMP ERROR        ;IR A RUTINA DE ERRORES.
12DB                   ;
12DB                   ;
12DB  A9 08     RESPON LDA #$08         ;OBTENER NUMERO DE DISCO.
12DD  20 B1 FF         JSR LISTEN       ;HACER QUE EL DISCO ESCUCHE.
12E0  A9 6F            LDA #$6F
12E2  4C 93 FF         JMP SECLIS       ;SECUNDARIO DESPUES DE ESCUCHAR.
12E5                   ;
```

Sigue →

Listado 1 continuación

```
12E5              ;
12E5  A9 08       SHUT    LDA #$08        ;NUMERO DEL FICHERO LOGICO.
12E7  4C C3 FF            JMP CLOSE       ;CERRAR EL FICHERO.
12EA              ;
12EA              ;
12EA  A9 49       ICMD    LDA #'I         ;COMANDO DE INICIALIZACION.
12EC  2C                  .BYTE $2C
12ED  A9 56       CCMD    LDA #'V         ;COMANDO DE VALIDEZ.
12EF  48                  PHA             ;SALVAR EL COMANDO.
12F0  20 DB 12            JSR RESPON      ;FIJAR CANAL DE COMANDOS.
12F3  68                  PLA             ;OBTENER COMANDO Y
12F4  20 A8 FF            JSR CIOUT       ;ENVIARLO.
12F7  4C AE FF            JMP UNLIST      ;PROVOCAR UN 'UN-LISTEN'
12FA              ;
12FA              ;
12FA  A9 4E       HDISK   LDA #'N         ;PREPARAR COMANDO "HEADER"
12FC  2C                  .BYTE $2C
12FD  A9 52       RDISK   LDA #'R         ;PREPARAR COMANDO "RENAME"
12FF  2C                  .BYTE $2C
1300  A9 53       SDISK   LDA #'S         ;PREPARAR COMANDO "SCRATCH"
1302  A2 00               LDX #$00
1304  9D 3C 03            STA CMDBUF,X    ;ALMACENAR COMANDO AQUI.
1307  E8                  INX
1308  A9 3A               LDA #':         ;DOS PUNTOS COMUNES PARA TODOS.
130A  9D 3C 03            STA CMDBUF,X    ;ALMACENARLOS AQUI.
130D  E8                  INX             ;PREPARAR PARA SIGUIENTE BYTE.
130E  60                  RTS
130F              ;
130F              ;
130F  A2 00       SURE    LDX #$00
1311  BD 3E 13    SPEAK   LDA MESSAG,X    ;ENVIAR MENSAJE:
1314  F0 06               BEQ SILENT      ;ESTAS SEGURO?
1316  20 42 E7            JSR CHROUT
1319  E8                  INX
131A  D0 F5               BNE SPEAK       ;BIFURCAR SIEMPRE.
131C  20 CF FF    SILENT  JSR INPUT       ;BUSCAR S=SI
131F  C9 53               CMP #'S
1321  08                  PHP
1322  20 36 13            JSR CARRET      ;IMPRIMIR RETORNO DE CARRO.
1325  28                  PLP
1326  60                  RTS
1327              ;
1327              ;
1327  20 E4 FF    PAUSE   JSR GETIN       ;OBTENER UN BYTE.
132A  F0 09               BEQ NOWAIT      ;NO HAY, SIGUE.
132C  C9 20               CMP #$20
132E  D0 05               BNE NOWAIT      ;NINGUN ESPACIO DISPONIBLE.
1330  20 E4 FF    KEYUP   JSR GETIN       ;AHORA ESPERAR
1333  F0 FB               BEQ KEYUP       ;OTRA PULSACION
1335  60          NOWAIT  RTS
1336              ;
1336              ;
1336  A9 0D       CARRET  LDA #$0D        ;IMPRIMIR RETORNO DE CARRO.
1338  2C                  .BYTE $2C
1339  A9 20       SPACE   LDA #$20        ;IMPRIMIR UN ESPACIO.
133B  4C D2 FF            JMP OUTPUT
133E              ;
133E              ;
133E              ;TABLA DE DATOS Y DIRECCIONES PARA DISK-O-VIC
133E              ;
133E              ;
133E  0D          MESSAG  .BYTE $0D
133F  45 53               .BYTE 'ESTAS SEGURO? (S/N)'
1353  00                  .BYTE $00
1354  4C          SYSTM1  .BYTE $4C       ;NEW CHRGET.
1355  13 10               .WORD INSERT
1357  00                  .BYTE $00
1358  B0          SYSTM2  .BYTE $B0, $0A  ;OLD CHRGET.
1359  0A
135A  C9                  .BYTE $C9, $00
135B  00
135C              ;
135C              ;
135C              ;TABLA DE PALABRAS CLAVE DE DISK-O-VIC
135C              ;
135C              ;
135C              ;NOTA. EL BYTE ADICIONAL DE CADA LINEA
135C              ;ES LA ULTIMA LETRA, EN POSICION DE "SHIFT"
135C              ;(ASCII   $80) PARA EL ANALIZADOR
135C              ;
135C              ;
135C  41 50       KEYWRD  .BYTE 'APPEN', $C4
1361  C4
1362  43 41               .BYTE 'CATALO', $C7
1368  C7
1369  43 4F               .BYTE 'COLLEC', $D4
136F  D4
1370  44 4C               .BYTE 'DLOA', $C4
1374  C4
1375  44 53               .BYTE 'DSAV', $C5
1379  C5
137A  48 45               .BYTE 'HEADE', $D2
137F  D2
1380  49 4E 49            .BYTE 'INI', $D4
1383  D4
1384  4B 49 4C            .BYTE 'KIL', $CC
1387  CC
1388  4F 46               .BYTE 'OF', $C6
138A  C6
138B  52 45               .BYTE 'RENAM', $C5
```

Dado que el programa está escrito en lenguaje máquina, hace falta un monitor de lenguaje máquina para introducirlo. El VIC-20 no dispone de un monitor residente, pero se ven cada vez más monitores de acoplamiento. Dos buenas opciones son el VICmon o el Tinymon.

El VICmon, fabricado por Commodore, es el monitor de lenguaje máquina oficial para el VIC-20 y ofrece muchos comandos. Se hace en forma de cartucho (cartridge) (ROM) y se encaja fácilmente en la puesta para ampliación. Por otro lado, el Tinymon, es un monitor de cinta o cargado en disco. La ventaja del Tinymon es que puede ser tecleado por el usuario y así se ahorra bastante dinero. No soporta tantos comandos como el VICmon pero eso no importa para el propósito que tenemos aquí. Sólo nos interesan los comandos de S (salvar) y de M (volcado de memoria). Por lo tanto, cualquier monitor sirve nuestros propósitos.

Para hacer una copia de DISK-O-VIC para el ordenador se siguen las instrucciones que se presentan a continuación:

—Desconectar cualquier acoplamiento adicional a la memoria. DISK-O-VIC tiene que introducirse en una máquina "stock".

—Cargar un monitor de lenguaje máquina. Sirve igual un monitor basado en cinta/disco o de "cartridge".

—Se introduce el código objeto mediante perforación utilizando como guía el Listado 2. Se inicia la introducción en la posición $1000 y se continúa hacia arriba.

—Una vez introducido el código, se modifican las siguientes posiciones. Introducir el byte de datos $2F en las posiciones $2D, $2F y $31. Se introduce el byte de datos $15 en las posiciones $2E, $30 y $32. Todas estas posiciones son de página cero.

—El monitor sale en Basic usando el comando X.

—Se salva el programa utilizando el comando normal de "SAVE" del VIC-20. El programa puede salvarse tanto en cinta como en disco.

—Si así se desea, cualquier acoplamiento adicional de memoria puede volver a conectarse.

Así se dispone de una versión completa de DISK-O-VIC lista para usarse. El código que se acaba de introducir y salvar es muy especial. Se puede cargar y ejecutar como cualquier otro programa de Basic. Cuando se ejecuta el programa, un cargador especial automáticamente vuelve a ubicar el DIS-O-VIC al principio de la memoria, sea cual sea la localización. Además, el cargador compensa de forma instantánea cualquier memoria adicional que se encuentre conectada al VIC-20.

```
1390  C5
1391  53 43           .BYTE 'SCRATC', $C8
1397  C8
1398  53 45 4E        .BYTE 'SEN', $C4
139B  C4
139C  53 54           .BYTE 'STATU', $D3
13A1  D3
13A2  00              .BYTE $00       ;FIN DE MARCADOR DE PALABRAS CLAVE
13A3                  ;
13A3                  ;
13A3                  ;*** DIRECCIONES DE ACCIONES DE RUTINA ***
13A3                  ;
13A3                  ;
13A3  98 10   ACTION  .WORD APPEND-1
13A5  B0 10           .WORD CAT-1
13A7  2D 11           .WORD COLLEC-1
13A9  33 11           .WORD DLOAD-1
13AB  61 11           .WORD DSAVE-1
13AD  81 11           .WORD HEADER-1
13AF  BE 11           .WORD INIT-1
13B1  C4 11           .WORD KILL-1
13B3  C7 11           .WORD OFF-1
13B5  CC 11           .WORD RENAME-1
13B7  16 12           .WORD SCRATC-1
13B9  29 12           .WORD SEND-1
13BB  4C 12           .WORD STATUS-1
13BD                  ;
13BD                  ;
13BD                  ;VARIABLES DE DISK-O-VIC
13BD                  ;
13BD                  ;
13BD  00      SSAVE   .BYTE $00       ;SALVAR REGISTRO DE ESTADO
13BE  00      ASAVE   .BYTE $00       ;SALVAR ACUMULADOR
13BF  00      XSAVE   .BYTE $00       ;SALVAR REGISTRO X
13C0  00      YSAVE   .BYTE $00       ;SALVAR REGISTRO Y
13C1                  .END
```

La tarea de teclear este programa puede resultar laboriosa, por lo cual es aconsejable que se realice entre varios usuarios. Es una ventaja que aunque el programa se ha escrito en lenguaje máquina, éste se parece a Basic para el VIC-20. Esto significa que es bastante fácil realizar copias de reserva. Se hace cargando el DISK-O-VIC (sin ejecutarlo) y se salvan más copias utilizando el comando normal de Save.

Conclusión

El valor práctico de DISK-O-VIC es evidente, pero el programa también debe servir como ejemplo de la forma en que un sistema operativo en disco completo puede ser implementado en el VIC-20. El ordenador evidentemente contiene muchas rutinas importantes en ROM y le incumbe a cada usuario aprender todo lo que puede sobre ellas. El programa también demuestra que la unidad de disco 1541 constituye una unidad enormemente flexible.

La programacón del VIC-20 y la 1541 en lenguaje máquina para que realicen comandos nuevos y más complicados no resulta tan difícil como parece al principio. La clave, por supuesto, es dividir el problema en una serie de subrutinas más pequeñas, aprovechando al máximo las numerosas rutinas ROM disponibles. Fue este el procedimiento utilizado en DISK-O-VIC.

Listado 2. El volcado hexadecimal del programa de utilidad DIS-O-VIC.
Nota: Se precisa un monitor de lenguaje de máquina para ejecutar este listado.

```
 PC   IRQ  SR AC XR YR SP
B780 E455 34 33 38 36 F8

1000 00 1C 10 01 00 99 22 93
1008 11 11 12 22 A3 36 29 22
1010 44 49 53 4B 2D 4F 2D 56
1018 49 43 22 00 3A 10 02 00
1020 99 22 11 20 44 49 53 4B
1028 20 43 4F 4D 4D 41 4E 44
1030 20 50 41 43 4B 41 47 45
1038 22 00 53 10 03 00 99 A3
1040 34 29 22 46 4F 52 20 54
1048 48 45 20 56 49 43 2D 32
1050 30 22 00 62 10 04 00 99
1058 A3 31 30 29 22 11 42 59
1060 22 00 7A 10 05 00 99 A3
1068 35 29 22 11 54 48 4F 4D
1070 41 53 20 48 45 4E 52 59
1078 22 00 93 10 06 00 9E 32
1080 35 36 AC C2 28 34 34 29
1088 AA C2 28 34 33 29 AA 31
1090 34 38 00 00 00 A5 2D 85
1098 22 A5 2E 85 23 A5 37 85
10A0 24 A5 38 85 25 A0 00 A5
10A8 22 D0 02 C6 23 C6 22 B1
10B0 22 D0 3C A5 22 D0 02 C6
10B8 23 C6 22 B1 22 F0 21 85
10C0 26 A5 22 D0 02 C6 23 C6
10C8 22 B1 22 18 65 24 AA A5
10D0 26 65 25 48 A5 37 D0 02
10D8 C6 38 C6 37 68 91 37 8A
10E0 48 A5 37 D0 02 C6 38 C6
10E8 37 68 91 37 18 90 B6 C9
10F0 BF D0 ED A5 37 85 33 A5
10F8 38 85 34 6C 37 00 BF A2
1100 00 00 A0 00 00 BD 93 FF
1108 00 F0 07 99 7E 00 00 C8
1110 E8 D0 F4 4C 74 C4 B0 0A
1118 C9 20 F0 32 38 E9 30 38
1120 E9 D0 8D FD FF 00 08 68
1128 8D FC FF 00 8E FE FF 00
1130 8C FF FF 00 BA BD 01 01
1138 C9 8C D0 07 BD 02 01 C9
1140 C4 F0 12 AC FF FF 00 AE
1148 FE FF 00 AD FC FF 00 48
1150 AD FD FF 00 28 60 4C 73
1158 00 00 AD FC FF 00 4A 90

1160 3A A2 00 00 86 0B A4 7A
1168 B9 00 00 02 38 FD 9B FF
1170 00 F0 13 C9 80 F0 13 E6
1178 0B E8 BD 9A FF 00 10 FA
1180 BD 9B FF 00 D0 E4 F0 C6
1188 E8 C8 D0 E0 84 7A A5 0B
1190 0A AA BD E3 FF 00 48 BD
1198 E2 FF 00 48 20 7B FC 00
11A0 4C 73 00 00 68 68 20 7B
11A8 FC 00 20 6B C9 4C 9F C4
11B0 A0 02 A5 2D D0 02 C6 2E
11B8 C6 2D 88 D0 F5 A5 2D 85
11C0 FB A5 2E 85 FC 4C 7B FD
11C8 00 20 29 FF 00 A9 24 8D
11D0 3C 03 A2 3C A0 03 A9 01
11D8 20 BD FF A9 0E A0 60 A2
11E0 08 20 BA FF 20 C0 FF A9
11E8 08 20 B4 FF A9 60 20 96
11F0 FF A9 00 00 85 90 A0 03
11F8 8C 3C 03 20 A5 FF 85 FD
1200 A4 90 D0 39 20 A5 FF 85
1208 FE A4 90 D0 30 AC 3C 03
1210 88 D0 E5 A6 FD A5 FE 20
1218 CD DD 20 78 FF 00 20 A5
1220 FF A6 90 D0 19 C9 00 00
1228 F0 0E 20 D2 FF 20 E1 FF
1230 F0 0D 20 66 FF 00 4C 42
1238 FD 00 20 75 FF 00 A0 02
1240 D0 BB A9 0E 20 C3 FF 20
1248 CC FF 4C 74 C4 20 2C FF
1250 00 4C 74 C4 A5 2B 85 FB
1258 A5 2C 85 FC 20 ED FE 00
1260 20 D8 FE 00 A9 00 00 A6
1268 FB A4 FC 20 D5 FF 20 33
1270 C5 A5 22 A6 23 18 69 02
1278 85 2D 90 01 E8 86 2E 20
1280 59 C6 4C 88 FD 00 20 ED
1288 FE 00 20 D8 FE 00 A5 2B
1290 85 FB A5 2C 85 FC A9 FB
1298 A6 2D A4 2E 20 D8 FF 20
12A0 24 FF 00 20 92 FE 00 4C
12A8 74 C4 20 39 FF 00 20 EF
12B0 FE 00 20 73 00 00 C9 2C
12B8 D0 2D 9D 3C 03 E8 20 73
12C0 00 00 C9 49 D0 22 20 73
12C8 00 00 F0 1D 9D 3C 03 E8
12D0 20 73 00 00 F0 14 9D 3C

12D8 03 E8 20 73 00 00 D0 0B
12E0 9D 3C 03 20 4E FF 00 D0
12E8 09 4C 71 FE 00 4C 08 CF
12F0 20 29 FF 00 4C 74 C4 20
12F8 22 FD A2 04 4C 41 FC 00
1300 20 3C FF 00 20 79 00 00
1308 F0 E7 C9 22 D0 E3 A5 7A
1310 85 FB A5 7B 85 FC 20 73
1318 00 00 F0 D6 C9 22 D0 F7
1320 20 73 00 00 C9 54 D0 CB
1328 20 73 00 00 C9 4F D0 C4
1330 20 73 00 00 20 EF FE 00
1338 A9 3D 9D 3C 03 E8 A5 FB
1340 85 7A A5 FC 85 7B 20 EF
1348 FE 00 A9 00 00 9D 3C 03
1350 4C 71 FE 00 20 3F FF 00
1358 20 EF FE 00 A9 00 00 9D
1360 3C 03 20 4E FF 00 D0 29
1368 4C 71 FE 00 20 ED FE 00
1370 A9 00 00 9D 3C 03 20 29
1378 FF 00 20 1A FF 00 A0 00
1380 00 B9 3C 03 F0 06 20 A8
1388 FF C8 90 F5 20 AE FF 20
1390 92 FE 00 F0 03 20 92 FE
1398 00 4C 74 C4 A9 08 85 BA
13A0 20 B4 FF A9 6F 20 96 FF
13A8 A0 00 00 20 A5 FF 99 3C
13B0 03 C8 C9 0D D0 F5 A9 00
13B8 00 99 3C 03 20 AB FF A0
13C0 00 00 A9 30 D9 3C 03 D0
13C8 06 C8 D9 3C 03 F0 15 20
13D0 24 FF 00 20 CC FF 20 75
13D8 FF 00 A9 3C A0 03 20 1E
13E0 CB 68 68 4C 74 C4 60 20
13E8 29 FF 00 BA A2 3C A0 03
13F0 20 BD FF A9 08 AA A0 00
13F8 00 84 90 4C BA FF A2 00
1400 00 86 B7 20 79 00 00 F0
1408 1F C9 22 D0 1B E6 7A D0
1410 02 E6 7B A0 00 00 B1 7A
1418 F0 0A C9 22 F0 06 9D 3C
1420 03 E8 D0 EA E4 B7 F0 01
1428 60 68 68 4C 08 CF A9 08
1430 20 B1 FF A9 6F 4C 93 FF
1438 A9 08 4C C3 FF A9 49 2C
1440 A9 56 4B 20 1A FF 00 68
1448 20 A8 FF 4C AE FF A9 4E

1450 2C A9 52 2C A9 53 A2 00
1458 00 9D 3C 03 E8 A9 3A 9D
1460 3C 03 E8 60 A2 00 00 BD
1468 7D FF 00 F0 06 20 42 E7
1470 E8 D0 F5 20 CF FF C9 53
1478 08 20 75 FF 00 28 60 20
1480 E4 FF F0 09 C9 20 D0 05
1488 20 E4 FF F0 FB 60 A9 0D
1490 2C A9 20 4C D2 FF 0D 45
1498 53 54 41 53 20 53 45 47
14A0 55 52 4F 3F 20 28 53 2F
14A8 4E 29 20 00 00 4C 52 FC
14B0 00 00 00 B0 0A C9 00 00
14B8 41 50 50 45 4E C4 43 41
14C0 54 41 4C 4F C7 43 4F 4C
14C8 4C 45 43 D4 44 4C 4F 41
14D0 C4 44 53 41 56 C5 48 45
14D8 41 44 45 D2 49 4E 49 D4
14E0 4B 49 4C CC 4F 46 C6 52
14E8 45 4E 41 4D C5 53 43 52
14F0 41 54 43 C8 53 45 4E C4
14F8 53 54 41 54 55 D3 00 00
1500 D7 FC 00 EF FC 00 6C FD
1508 00 72 FD 00 A0 FD 00 C0
1510 FD 00 FD FD 00 03 FE 00
1518 06 FE 00 0B FE 00 55 FE
1520 00 68 FE 00 BB FE 00 00
1528 00 00 00 00 00 00 00 AA
```

VIDEO CASINO

Batalla Naval

Estrategia de Vida o Muerte en Alta Mar.



Si te gusta la estrategia naval o simplemente quieres mejorar tu puntería, intenta navegar por los mares de este juego.

VIC 20

COMMODORE 64

El juego "Batalla Naval", para un jugador en el VIC-20 no ampliado, se basa en el popular juego de "Barquitos". El jugador tiene que hundir los barcos que pertenecen al ordenador antes de que éste hunda los del jugador.

El jugador y el ordenador tienen al principio del juego, cinco barcos cada uno, colocados sobre un tablero. A continuación, se presentan los nombres de los barcos y el número de veces que hay que alcanzarlos para que se hundan:

—Portaaviones	5 veces
—Acorazado	4 veces
—Crucero	3 veces
—Submarino	3 veces
—Destructor	2 veces

Al iniciar el juego el jugador tendrá que elegir entre unos niveles de dificultad del uno al tres. En el nivel uno, el jugador y el ordenador disparan una vez cada uno. En el nivel dos, el ordenador dispara dos veces y el jugador una. En el nivel tres, es casi imposible que gane el jugador, ya que el ordenador dispara tres veces y el jugador una. La única oportunidad que tiene el jugador de ganar es tener suerte y poder encontrar los cinco barcos enemigos en seguida. El ordenador dispara al azar, pero nunca da dos veces en el mismo sitio.

Después de elegir el nivel de dificultad, el jugador coloca sus barcos en el tablero. El ordenador le pide al jugador que le dé una coordenada inicial para el Portaaviones. El jugador introduce una coordenada con el siguiente formato: una letra de A a J seguida por un número de 0 a 9 (por ejemplo, D7 o J0). A continuación, el ordenador le pide al jugador que entre una H o una V para indicar si el barco se tiene que colocar en posición horizontal o vertical.

Si el jugador da una coordenada no correcta, o intenta colocar un barco encima de otro, o trata de situar un barco de tal forma que se sale del tablero, el programa le dirá "Mala Coordenada" y volverá a empezar. Las coordenadas del barco se dibujan

en el tablero a medida que vayan introduciéndose.

Después que el jugador haya colocado sus barcos, el ordenador situará los suyos. El jugador no los puede ver, dado que se representan con caracteres programables que tienen el mismo aspecto que los que forman el tablero.

La Batalla Comienza

Cuando el ordenador termina de situar sus barcos, le pide al jugador que le indique las coordenadas del primer disparo. A continuación, dispara el ordenador. Tanto los disparos acertados como los fallados se representan en el tablero. Si se hunden primero todos los barcos del jugador, el ordenador dice "¡Gano Yo!" y le enseña la posición de sus barcos al jugador. Cuando el jugador alcanza o

La Tabla 1. Una idea general del listado de programa para "Batalla Naval"

Líneas	
50	Bajar parte superior de memoria y realizar procesos aleatorios
60-80	Establecer caracteres programables
90	Fijar colores: indicar al ordenador que busque la información de caracteres en la RAM
100-120	Fijar nivel de dificultad
130	Inicializar variables
140-200	Dibujar representación en pantalla
210-360	Colocar barcos amigos
370-440	Colocar barcos enemigos
450-580	Jugador dispara a los barcos del ordenador
590-670	Dispara ordenador
680-700	Representar mensajes de barco alcanzado y barco hundido
710-750	Fin de rutina de juego
760	Borrar parte superior de pantalla
770	Mensaje de mala coordenada
780	Cronometrar representación de mensaje y luego borrar
790	Rutina de sonido para los disparos
800	Borrar el tablero del enemigo
810-820	Rutina de sonido para blanco alcanzado
830	Representar blancos alcanzados
840	Nombres de barcos y disparos que faltan para hundir barco.

Listado 1. Batalla Naval VIC-20

```
50 POKE52,28:POKE56,28:CLR:POKE143,VAL(MID$(TI$,5,2))
60 FORI=7176TO7679:POKEI,PEEK(I+25600):NEXT
70 FORI=7168TO7174:POKEI,1:NEXT:POKE7175,255
80 FORI=7632TO7664STEP8:FORJ=0TO7:POKEI+J,PEEK(7168+J):NEXT:NEXT
90 PRINTCHR$(147)CHR$(5):POKE36879,110:POKE36869,255
100 PRINT:PRINTTAB(7)"ELEGIR":PRINT:PRINT"[SPC]NIVEL[SPC]DE[SPC]DIFICULTAD"
110 PRINT:PRINT"[SPC]FACIL[SPC](1-3)[SPC]DIFICIL[2SPC]":INPUTU
120 IFU<1ORU>3THEN90
130 W=30720:S1=36876:S2=36877:V=36878:P=1
140 PRINTCHR$(147)::FORI=1TO8:PRINT:NEXT
150 PRINT"[SPC]0123456789[SPC]0123456789";
160 FORI=7900TO8119:POKEI,0:NEXT
170 FORI=7900TO8098STEP22:POKEI,P:POKEI+11,P:P=P+1:NEXT
180 FORI=1TO10:PRINT:NEXT
190 PRINTCHR$(158)"--BARCOS-----BARCOS---";
200 PRINT"--AMIGOS-----ENEMIGOS-";
210 FC=7901:FORR=1TO5:READS$(R),SR(R)
220 PRINTCHR$(19)"[SPC]ENTRAR[SPC]COORDENADAS"
230 PRINT"DEL[SPC]"S$(R):INPUTC$
240 PRINT"[SPC]H[SPC]PARA[SPC]HORIZONTAL":PRINT"[SPC]V[SPC]PARA[SPC]VERTICAL":IN
PUTD$
250 IFD$="H"THEND=1:GOTO280
260 IFD$<>"V"THENGOSUB760:GOTO770
270 D=22
280 GOSUB760
290 X=VAL(RIGHT$(C$,1)):V$=RIGHT$(C$,1)
300 IFASC(V$)<48ORASC(V$)>57THEN770
310 Y=ASC(C$)-65:IFY>9THEN770
320 IFLEN(C$)>2THEN770
330 FS=FC+Y*22+X:FORSC=1TOSR(R)
340 IFPEEK(FS)<>0THEN770
350 POKEFS+W,7:POKEFS,35
360 FS=FS+D:NEXTSC:NEXTR
370 PRINTCHR$(19):PRINT:PRINT"[SPC]ESTOY[SPC]COLOCANDO[SPC]MIS[3SPC]BARCOS."
380 POKEV,15:EC=7912:MS=58:FORR=1TO5
390 CS=INT(RND(1)*20)+230:POKES1,CS
400 X=INT(RND(1)*10):Y=INT(RND(1)*10):D=INT(RND(1)*2)+1
410 IFD=2THEND=22
420 ES=EC+Y*22+X:FORSC=1TOSR(R)
430 IFPEEK(ES)<>0THEN800
440 POKEES,MS:ES=ES+D:NEXTSC:MS=MS+1:NEXTR:POKES1,0
450 GOSUB760:PRINTCHR$(19):PRINT"[SPC]ENTRA[SPC]COORDENADAS[SPC]Y[2SPC]RETURN[SP
C]PARA[SPC]DISPARAR"
460 FF=7912:INPUTF$:X=VAL(RIGHT$(F$,1)):V$=RIGHT$(F$,1)
470 IFASC(V$)<48ORASC(V$)>57THEN450
480 Y=ASC(F$)-65:IFY>9THEN450
490 IFLEN(F$)<>2THEN450
500 GOSUB760:GOSUB790:EF=FF+Y*22+X:PE=PEEK(EF):Q=0
510 IFPE=0ORPE=42THENGOSUB830:POKEEF,42:GOTO580
520 IFPE=58THENCA=CA+1:Q=1:HI=5:HS=CA
530 IFPE=59THENBA=BA+1:Q=2:HI=4:HS=BA
540 IFPE=60THENCR=CR+1:Q=3:HI=3:HS=CR
550 IFPE=61THENSU=SU+1:Q=4:HI=3:HS=SU
560 IFPE=62THENDE=DE+1:Q=5:HI=2:HS=DE
570 GOSUB680:IFEH=17THEN710
580 FORZ=1TOU:FORT=1TO500:NEXT
590 FF=7901:Y=INT(RND(1)*10):X=INT(RND(1)*10)
600 FX=FF+Y*22+X:IFPEEK(FX)=42THEN590
610 GOSUB790:IFPEEK(FX)=0THENEF=FX:GOSUB830:GOTO670
620 IFPEEK(FX)=35THENPOKEFX+W,2:EF=FX:GOSUB830:POKEFX,42:FH=FH+1
630 IFFH<>17THEN600
640 PRINT"GANO[SPC]YO[SPC]!!"
645 FORI=7912TO8185:IFPEEK(I)>57ANDPEEK(I)<63THENPOKEI+W,3:POKEI,35
650 NEXT:FORT=1TO5000:NEXT:GOTO720
660 NEXTZ:GOTO450
670 POKEFX,42:NEXTZ:GOTO450
680 POKEEF+W,2:GOSUB830:POKEEF,42
685 PRINTS$(Q)"[SPC]TOCADO":EH=EH+1:SS=2:GOSUB810:GOSUB780
690 IFHI=HSTHENPRINTS$(Q)"[SPC]HUNDIDO":SS=4:GOSUB810:GOSUB780
700 RETURN
710 PRINT"TU[SPC]GANAS[SPC]!!":SS=8:GOSUB810
720 FORT=1TO2000:NEXT:PRINTCHR$(147):PRINT"OTRA[SPC]VEZ?[SPC](S/N)"
730 GETPA$:IFPA$=""THEN730
740 IFPA$="S"THENCLR:GOTO90
750 END
760 FORI=7680TO7810:POKEI,32:NEXT:RETURN
770 PRINT"MALA[SPC]COORDENADA.[6SPC]VUELVE[SPC]A[SPC]EMPEZAR.":FORT=1TO2500:NEXT
:CLR:GOTO90
780 FORT=1TO1500:NEXT:GOSUB760:RETURN
790 POKES2,200:FORT=15TO0STEP-.1:POKEV,T:NEXT:POKES2,0:RETURN
800 FORG=1TO10:FORH=1TO10:POKEEC+EB,0:EB=EB+1:NEXT:EC=EC+22:EB=0:NEXT:GOTO380
810 POKEV,15:FORI=1TOSS:FORJ=1TO10
815 POKES1,230:FORT=1TO10:NEXT:POKES1,235:FORT=1TO10:NEXT
820 NEXT:POKES1,0:FORT=1TO50:NEXT:NEXT:RETURN
830 FORJ=1TO10:POKEEF,42:FORT=1TO20:NEXT:POKEEF,0:FORT=1TO20:NEXT:NEXT:RETURN
840 DATAPORTA-AVIONES,5,ACORAZADO,4,CRUCERO,3,SUBMARINO,3,DESTRUCTOR,2
```

Listado 2. Batalla Naval C-64

```
10 PRINTCHR$(147):FORI=1TO9:PRINT:NEXT
20 PRINTTAB(14)CHR$(13)"-SEA[SPC]BATTLE-":FORT=1TO1500:NEXT
30 FORI=1TO6:PRINT:NEXT:PRINTTAB(16)"POR[SPC]DAVE[SPC]GARDNER"
40 PRINT:PRINT"[SPC]ESPERE[SPC]POR[SPC]FAVOR..."
50 POKE52,48:POKE56,48:CLR:POKE143,VAL(MID$(TI$,5,2))
56 POKE56334,PEEK(56334)AND254
58 POKE1,PEEK(1)AND251
60 FORI=0TO511:POKEI+12288,PEEK(I+53248):NEXT
62 POKE1,PEEK(1)OR4
64 POKE56334,PEEK(56334)OR1
70 FORI=12288TO12294:POKEI,1:NEXT:POKE12295,255
80 FORI=12752TO12784STEP8:FORJ=0TO7:POKEI+J,PEEK(12288+J):NEXT:NEXT
90 PRINTCHR$(147)CHR$(5):POKE53280,12:POKE53281,12
92 POKE53272,(PEEK(53272)AND240)+12
100 PRINT:PRINTTAB(7)"ELEGIR":PRINT:PRINT"[SPC]NIVEL[SPC]DE[SPC]DIFICULTAD"
110 PRINT:PRINT"[SPC]FACIL[SPC](1-3)[SPC]DIFICIL[2SPC]":INPUTU
120 IFU<1ORU>3THEN90
130 W=54272:P=1:WF=W+4:V=W+24:AT=W+5:SE=W+6:HF=W+1:LF=W:POKEV,15
140 PRINTCHR$(147):FORI=1TO8:PRINT:NEXT
150 PRINTTAB(10)"0123456789[SPC]0123456789"
160 FORI=1433TO1793STEP40:FORJ=0TO21:POKEI+J,0:POKEI+W+J,1:NEXT:NEXT
170 FORI=1433TO1793STEP40:POKEI,P:POKEI+11,P:P=P+1
175 NEXT
180 FORI=1TO10:PRINT:NEXT
190 PRINTCHR$(5)TAB(10)"--BARCOS-----BARCOS---"
200 PRINTTAB(10)"--AMIGOS-----ENEMIGOS-";
210 FC=1434:FORR=1TO5:READS$(R),SR(R)
220 PRINTCHR$(19)"[SPC]ENTRAR[SPC]COORDENADAS"
230 PRINT"DEL[SPC]"S$(R):I'PUTC$
240 PRINT"[SPC]H[SPC]PARA[SPC]HORIZONTAL":PRINT"[SPC]V[SPC]PARA[SPC]VERTICAL":IN
PUTD$
250 I:D$="H"THEND=1:GOTO280
260 IFD$<>"V"THENGOSUB760:GOTO770
270 D=40
280 GOSUB760
290 X=VAL(RIGHT$(C$,1)):V$=RIGHT$(C$,1)
300 IFASC(V$)<48ORASC(V$)>57THEN770
310 Y=ASC(C$)-65:IFY>9THEN770
320 IFLEN(C$)>2THEN770
330 FS=FC+((Y*40)+X):FORSC=1TOSR(R)
340 IFPEEK(FS)<>0THEN770
350 POKEFS+W,7:POKEFS,35
360 FS=FS+D:NEXTSC:NEXTR
370 PRINTCHR$(19):PRINT:PRINT"[SPC]ESTOY[SPC]COLOCANDO[SPC]MIS[3SPC]BARCOS."
380 EC=1445:MS=58:FORR=1TO5
390 GOSUB850
400 X=INT(RND(1)*10)+0:Y=INT(RND(1)*10)+0:D=INT(RND(1)*2)+1
410 IFD=2THEND=40
420 ES=EC+((Y*40)+X):FORSC=1TOSR(R)
430 IFPEEK(ES)<>0THEN800
440 POKEES,MS:ES=ES+D:NEXTSC:MS=MS+1:NEXTR
450 GOSUB760:PRINTCHR$(19):PRINT"ENTRA[SPC]COORDENADAS[SPC]Y[SPC]RETURN[SPC]PARA
[SPC]DISPARAR"
460 FF=1445:INPUTF$:X=VAL(RIGHT$(F$,1))
465 V$=RIGHT$(F$,1)
470 IFASC(V$)<48ORASC(V$)>57THEN450
480 Y=ASC(F$)-65:IFY>9THEN450
490 IFLEN(F$)<>2THEN450
500 GOSUB760:GOSUB790:EF=FF+((Y*40)+X):PE=PEEK(EF):O=0
510 IFPE=0ORPE=42THENGOSUB830:POKEEF,42:GOTO580
520 IFPE=58THENCA=CA+1:O=1:HI=5:HS=CA
530 IFPE=59THENBA=BA+1:O=2:HI=4:HS=BA
540 IFPE=60THENCR=CR+1:O=3:HI=3:HS=CR
550 IFPE=61THENSU=SU+1:O=4:HI=3:HS=SU
560 IFPE=62THENDE=DE+1:O=5:HI=2:HS=DE
570 GOSUB680:IFEH=17THEN710
580 FORZ=1TOU:FORT=1TO500:NEXT
590 FF=1434:Y=INT(RND(1)*10)+0:X=INT(RND(1)*10)+0
600 FX=FF+((Y*40)+X):IFPEEK(FX)=42THEN590
610 GOSUB790:IFPEEK(FX)=0THENEF=FX:GOSUB830:GOTO670
620 IFPEEK(FX)=35THENPOKEFX+W,2:EF=FX:GOSUB830:POKEFX,42:FH=FH+1
630 IFFH<>17THEN600
640 PRINT"GANO[SPC]YO[SPC]!!"
642 FORI=1445TO1823:IFPEEK(I)>57ANDPEEK(I)<63THENPOKEI+W,3:POKEI,35
650 NEXT:FORT=1TO5000:NEXT:GOTO720
660 NEXTZ:GOTO450
670 POKEFX,42:NEXTZ:GOTO450
680 POKEEF+W,2:GOSUB830:POKEEF,42
685 PRINTS$(O)"[SPC]TOCADO":EH=EH+1:SS=2
687 GOSUB810:GOSUB780
690 IFHI=HSTHENPRINTS$(O)"[SPC]HUNDIDO":SS=4:GOSUB810:GOSUB780
700 RETURN
710 PRINT"TU[SPC]GANAS[SPC]!!":SS=8:GOSUB810
720 FORT=1TO2000:NEXT:PRINTCHR$(147):PRINT"OTRA[SPC]VEZ?[SPC](S/N)"
730 GETPA$:IFPA$=""THEN730
740 IFPA$="S"THENCLR:GOTO90
750 END
760 FORI=1024TO1265:POKEI,32:NEXT:RETURN
770 PRINT"MALA[SPC]COORDENADA.[6SPC]VUELVE[SPC]A[SPC]EMPEZAR.":FORT=1TO2500:NEXT
:CLR:GOTO90
780 FORT=1TO1500:NEXT:GOSUB760:RETURN
790 POKEAT,9:POKES,0:POKEHF,3:POKELF,244:POKEWF,129:FORT=1TO600:NEXT
792 POKEWF,128:GOSUB860:RETURN
800 FORG=1TO10:FORH=1TO10:POKEEC+EB,0:EB=EB+1:NEXT:EC=EC+40:EB=0:NEXT:GOTO380
810 FORI=1TOSS:POKEAT,9:POKESE,0:POKEHF,71:POKELF,12:POKEWF,17:FORT=1TO100
820 NEXT:POKEWF,16:NEXT:GOSUB860:RETURN
830 FORJ=1TO10:POKEEF,42:FORT=1TO20:NEXT:POKEEF,0:FORT=1TO20:NEXT:NEXT:RETURN
840 DATAPORTA-AVIONES,5,ACORAZADO,4,CRUCERO,3,SUBMARINO,3,DESTRUCTOR,2
850 CS=INT(RND(1)*40)+210:POKEAT,9:POKESE,0:POKEHF,CS-100:POKELF,CS:POKEWF,17
852 FORT=1TO50:NEXT:POKEWF,16
860 FORJ=WTOW+23:POKEJ,0:NEXT:RETURN
```

hunde uno de los barcos del ordenador, éste le indica cuál es.

Una característica interesante de este juego es la forma en que el ordenador sigue la pista de la colocación de los barcos. Se utiliza la memoria de pantalla para almacenar la información. El programa le permite al ordenador hacer un "PEEK" solamente en las posiciones que han sido alcanzadas por un disparo. Por lo tanto, el ordenador no puede hacer trampas mirando la pantalla.

Dado que los barcos tienen diferentes códigos de pantalla, el ordenador sabe cuándo un disparo ha dado en el blanco y qué barco ha sido alcanzado. Esto demuestra que la memoria de pantalla realmente se localiza en la RAM y que puede ser utilizada para el almacenamiento de datos además de la representación de caracteres.

5K es Suficiente

Yo escribí este programa a raíz de que un amigo me dijo que no se podía hacer. El se había comprado un ordenador que no era de Commodore y para el cual estaba escribiendo una versión de este juego. Se tropezó con un pequeño problema de programación, y yo le comenté que no hubiera tenido dicho problema si se hubiera comprado un VIC-20.

El opinaba que el 5K de memoria RAM del VIC no sería suficiente para este juego. A mí se me presentó una buena oportunidad para saber exactamente la capacidad de los 3583 bytes disponibles en el VIC. Yo creo que este juego demuestra que se puede hacer mucho dentro de las limitaciones del VIC. Aunque ahora dispongo de una ampliación de memoria de 16K, sigo realizando el 90 por ciento de mi trabajo en el 5K de memoria original.

club commodore

Compartiendo Experiencias entre amigos

Esta sección está dedicada a la colaboración de todos nuestros lectores y está dividida en dos partes:

1) Programación:

Programas y similares

2) Magia:

Trucos, sugerencias, etc.

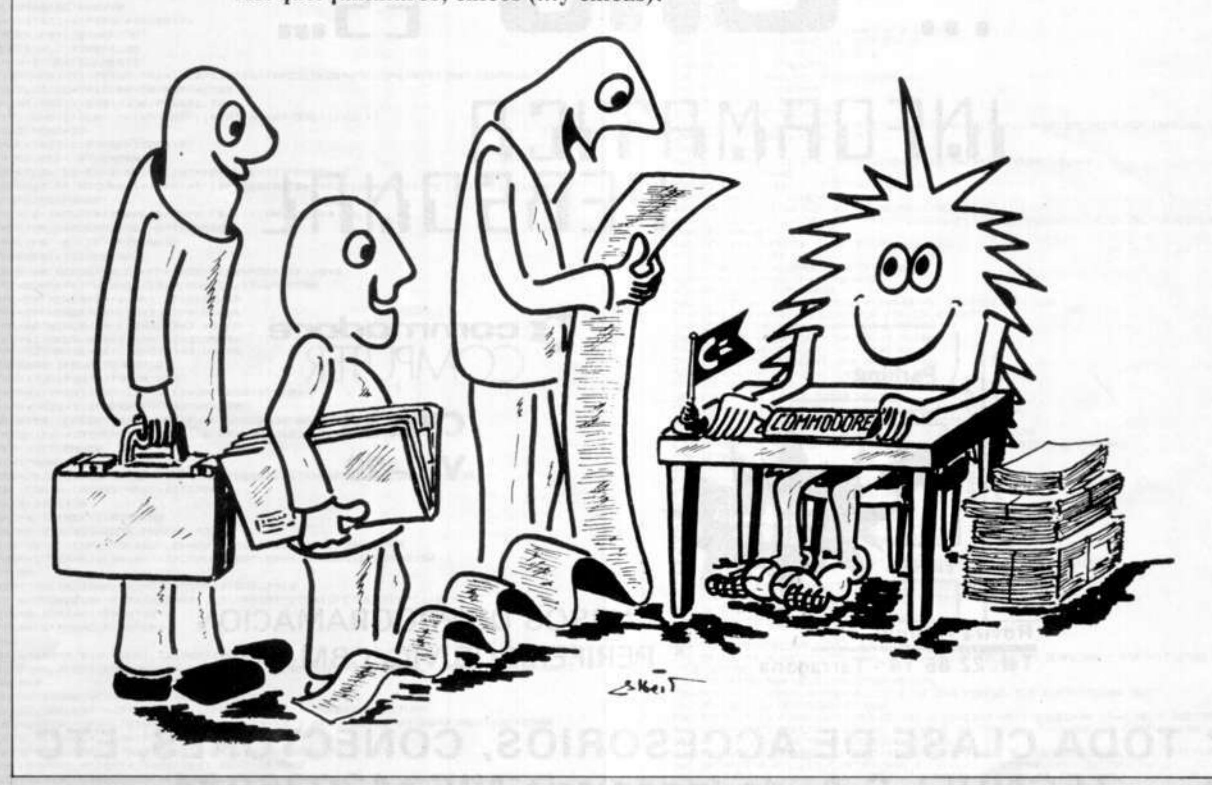
Habrá premios y alicientes "para todos los participantes" (ver editorial página 3).

Todas las colaboraciones deben venir escritas a máquina a doble espacio y los programas grabados en cinta (si es posible), o con el listado completo en impresora. Nuestros lectores más jóvenes pueden escribir a mano **pero con letra muy clara.**

Enviarnos vuestra dirección para que podáis poneros en contacto unos con otros.

Incluimos hoy dos colaboraciones de dos chavales a ver si a los demás se les quita el miedo y nos envían cosas. Topo Loco ya apareció anteriormente en Club Commodore.

Así que: ¡animaros, chicos (...y chicas)!





"Topo Loco"

Manuel Torralba
13 años
mándanos dirección

¡Hola!

Soy un vic-cioso de Barcelona. Recibo "Club Commodore"en mi casa cada mes y, como pedís colaboraciones, os mando este programa de juegos.

Mi nombre es Manuel Torralba y tengo 13 años. (La carta la ha pasado a máquina mi hermano mayor, que ha corregido todas (o casi todas) las faltas y ha puesto el rollo chulo. EL PROGRAMA ES MIO, LO HE HECHO YO SOLITO Y NADIE ME HA AYUDADO. (¡Es que nadie se cree que yo pueda programar así!).

El juego, que va en la cassette que os adjunto, se llama "Topo Loco". No necesita ninguna ampliación de memoria. El jugador debe "limpiar" el huerto (la pantalla) de lechugas (representadas por tréboles) sin chocar con los bloques grises que hay por todo el huerto, ni tampoco con el rastro que deja. Cuando acabas un huerto pasas a otro, y así sucesivamente hasta agotar los tres topos que posees. Para hacerlo más difícil, la velocidad del topo se incrementa poco a poco y, por cada huerto, el número de bloques aumenta en cinco. Y además, aunque me parecía difícil, le añadí música, color y sonido.

A continuación os explico el programa:

Línea 1:

Define el número de topos y huertos y continúa en la línea 3000.

Línea 8 a 9:

Ponen en pantalla el número de huertos.

Línea 10:

Define el número de lechugas.

LISTADO

```
PROGRAMA:10TOPO1

0 REM***************

1 REM*  TOPO LOCO  *

2 REM*-------------*

3 REM*   MANUEL    *

4 REM*             *

5 REM*  TORRALBA   *

6 REM***************

7 MN=1:GUS=3:GOSUB3000

8 POKE36879,8:FORT=1TO5:FORR=1TO210
:NEXTR:PRINT"[ CLR ]"

9 FORR=1TO210:NEXT:PRINT"[HOME ][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD][CRSRD]
[CRSRR][CRSRR][CRSRR][CRSRR]HUERTO"
;MN:NEXTT

10 LECH=15:POKE36879,8:PRINT"[ CLR ]
":POKE36878,15

11 POKE36879,8:PRINT"[ CLR ]":FORL=
1TO15+P1:POKE7680+INT(RND(1)*506),1
02:NEXTL

12 FORL=1TOLECH:POKE7680+INT(RND(1)
*506),88

13 POKE36878,15:POKE36876,0:FORE=1T
O120:NEXTE:POKE36876,235+L:NEXTL

14 FORE=1TO200:NEXTE

20 B=D:M=0:X=10:Y=10:C=255

25 GETTECLA$

30 IFTECLA$=""THENGOTO80

40 IFTECLA$=" "THENM=1

50 IFTECLA$="I"THENM=2

60 IFTECLA$="J"THENM=0

70 IFTECLA$="L"THENM=3

80 IFM=0THENX=X-1

81 POKE36878,15:POKE36876,190

90 IFM=1THENY=Y+1

100 IFM=2THENY=Y-1

110 IFM=3THENX=X+1

120 IFPEEK(7680+22*Y+X)<>88ANDPEEK(
7680+22*Y+X)<>32THEN GOTO150

121 IFPEEK(7680+22*Y+X)<>88THEN GOT
O130

122 POKE36878,15:FORN=150TO250:POKE
36876,N:NEXTN:POKE36874,0:SC=SC+20:
LECH=LECH-1

123 IFLECH=0THENGOTO9000

130 POKE7680+22*Y+X,81

135 FORA=1TOB:NEXTA:B=B-0.2

140 POKE36876,0:GOTO25

150 POKE7680+22*Y+X,42

155 POKE36878,15

160 POKE36876,0

161 FOR A=15TO0STEP-0.081:C=C-1:POK
E36878,A:POKE36879,C:POKE36877,180:
NEXTA:POKE36877,0

162 GUS=GUS-1

163 IFGUS=0THEN9019

168 GOTO11

190 POKE36878,15

1015 FORR=1TO100:NEXT

3000 PRINT"[ CLR ]":POKE36879,8:POK
E36865,140

3010 PRINT"[CRSRD][CRSRD][ WHT ][CRSRR]
[CRSRR]**TOPO LOCO**  ":PRINT"[CRSRD]
[CRSRD][CRSRR]MUEVE AL TOPO LOCO "

3011 PRINT"[CRSRD][CRSRR]Y COME TOD
AS":PRINT"[CRSRD][CRSRR]LAS LECHUGA
S([SHIFX])   "

3012 PRINT"[CRSRD][CRSRR]EVITANDO C
HOCAR"

3020 PRINT"[CRSRD]CON TU RASTRO O L
OS[CRSRR][CRSRR][CRSRR][CRSRD]BLOQU
ES GRISES"

3021 PRINT"[CRSRD][CRSRR]DIRIGE EL
TOPO CON"

3022 PRINT"[CRSRD][CRSRR] I,J,L,[RVSON]
SPACE[RVSOF]"

3030 PRINT"[CRSRD][CRSRR]  POR M.TO
RRALBA "

3060 FORALTURA=140 TO 40STEP-0.11:P
OKE36865,ALTURA:NEXTALTURA

3090 FORR1=1TO3000:NEXTR1:POKE198,0

3091 PRINT"[ CLR ][CRSRD][CRSRD][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD](1 ES E
L MAS DIFICIL)"

4000 INPUT"[HOME ][ WHT ][CRSRD][CRSRD]
[CRSRD][CRSRD]DIFICULTAD (1-10)";D

4010 D=D*15 :RETURN

9000 POKE36878,15

9011 FORR=1TO34 :READT:POKE36876,T:
POKE36879,T:FORM=1TO150:NEXTM:NEXTR
:RESTORE

9012 P1=P1+5:MN=MN+1:GOTO8

9019 POKE36878,15

9020 FORU=1TO34:READA:NEXTU

9021 POKE36879,8:FORR=1TO19 :READT:
POKE36876,T:FORM=1TO150:NEXTM:NEXTR

9022 PRINT"[HOME ][CRSRR][CRSRR][CRSRR]
[CRSRR][CRSRR][CRSRR][CRSRR][CRSRD]
[CRSRD][CRSRD][CRSRD][CRSRD][CRSRD]
[CRSRD]GAME OVER":FORR=1TO3000:NEXT
R:RUN

20000 DATA0,0,8,195,209,209,215,219
,209,219,215

20010 DATA195,209,209,215,219,209,2
09,207,209,209,215,219,221,219,215

20020 DATA209,207,195,201,207,209,2
09,8,8

20030 DATA0,0,225,223,0,219,215,0,2
09,207,201,207,201,195,0,0,225,0

READY.
```



Línea 11 y 12:
Colocan las lechugas y los bloques en su lugar.

Línea 20:
Define velocidad de juego, posición XY y la variable de color.

Línea 25 a 110:
Movimiento del topo.

Línea 120 y 121:
Comprueban qué objeto tiene delante el topo. Si es un bloque, pasa a la línea 150.

Línea 122:
El topo ha encontrado una lechuga.

Línea 123:
Caso de 0 lechugas, pasa a la línea 9000.

Línea 130:
Pone al topo en su posición.

Línea 135:
Retardo (velocidad).

Línea 140:
Vuelve a la línea 25.

Línea 150 a 168:
Explosión y un topo menos. Caso de 0 topos, pasan a la línea 9019.

Línea 3000 a 3090:
Instrucciones.

Línea 4000 a 4010:
Piden nivel dificultad.

Línea 9000 a 9012:
Música éxito (huerto vacío).

Línea 9019 a 9021:
Música de fin de juego.

Línea 9022:
Mensaje fin de juego.

Línea 20000 a 20030:
Datas musicales.

Principales variables: número de huertos (mn), número de topos (gus), número de lechugas (lech), velocidad (b), bloques adicionales (pl), posición del topo (X e Y)

Esperando noticias vuestras, se despide un colaborador.

Manel

Funny Faces

Esto no os lo explico porque es tan tonto, tan tonto, tan tonto...! Que os divirtáis con los muñecos que hacen cosas.

Iñigo Fry
Maestro Chapi, 15
MADRID-16 **14 años**







```
10 PRINT"[CLR][9CRSRD]"
20 PRINT"[10SPC][SHIFTU][4COMM+][SHIFTI]"
30 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB]"
40 PRINT"[10SPC][SHIFTB]-[2SPC]O[SHIFTB]"
50 PRINT"[10SPC][SHIFTB][2SPC][COMMG][SPC][SHIFTB]"
60 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB]"
70 PRINT"[10SPC][SHIFTB][SHIFTM][2COMMP][SHIFTN][SHIFTB]"
75 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB]"
80 PRINT"[10SPC][SHIFTJ][SHIFTC][COMMS][COMMA][SHIFTC][SHIFTK]"
90 PRINT"[12SPC][SHIFT@][SHIFTL][2SPC]"
100 PRINT"[12SPC][SHIFTN][SHIFTM][2SPC]"
110 FORY=0TO500
120 PRINT"[HOM][12CRSRD][11CRSRR]-"
130 FORT=0TO200:NEXTT
140 PRINT"[HOM][12CRSRD][11CRSRR]O"
150 FORT=0TO1500:NEXTT
155 GOSUB200
160 NEXTY
200 PRINT"[HOM][16CRSRD][12CRSRR][SHIFTJ][SHIFTK]"
210 FORT=0TO400:NEXTT
220 PRINT"[HOM][16CRSRD][12CRSRR][2SPC]"
230 FORT=0TO1500:NEXTT
235 GOSUB300
240 RETURN
300 PRINT"[HOM][6CRSRD]"
305 PRINT"[15SPC][SHIFTN][SHIFTM][2SPC]"
310 PRINT"[13SPC][SHIFTM][SHIFTN][2SPC][SHIFTM][SPC]"
317 PRINT"[14SPC][SHIFTM][2SPC][SHIFTN][SPC]"
320 PRINT"[10SPC][SHIFTU][4COMM+][SHIFTM][SHIFTN][2SPC]"
330 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB][SHIFTM][2SPC]"
340 PRINT"[10SPC][SHIFTB]O[2SPC]O[SHIFTB][SPC]*[SHIFTI]"
350 PRINT"[10SPC][SHIFTB][2SPC][COMMG][SPC][SHIFTB][2SPC][SHIFTG]"
360 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB][2SPC][SHIFTT]"
370 PRINT"[10SPC][SHIFTB][SHIFTM][2COMMP][SHIFTN][SHIFTB][2SPC][COMMG]"
375 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB][SPC][SHIFTN][SPC]"
380 PRINT"[10SPC][SHIFTJ][SHIFTC][COMMS][COMMA][SHIFTC][SHIFTK][SHIFTN][2SPC]"
390 PRINT"[12SPC][SHIFT@][SHIFTL][COMMO][SHIFTN][3SPC]"
395 PRINT"[12SPC][SHIFTN][SHIFTM][5SPC]"
400 FORT=0TO1700:NEXTT
500 PRINT"[HOM][6CRSRD]"
505 PRINT"[19SPC]"
516 PRINT"[11SPC][4COMM@][4SPC]"
517 PRINT"[11SPC][COMMG][2SPC][COMMM][4SPC]"
520 PRINT"[9SPC][COMMP][COMM@][SHIFTL][2COMMP][SHIFT@][COMM@][COMMP][2SPC]"
530 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB][3SPC]"
540 PRINT"[10SPC][SHIFTB]O[2SPC]O[SHIFTB][3SPC]"
550 PRINT"[10SPC][SHIFTB][2SPC][COMMG][SPC][SHIFTB][3SPC]"
560 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB][3SPC]"
570 PRINT"[10SPC][SHIFTB][SHIFTM][2COMMP][SHIFTN][SHIFTB][3SPC]"
575 PRINT"[10SPC][SHIFTB][4SPC][SHIFTB][3SPC]"
580 PRINT"[10SPC][SHIFTJ][SHIFTC][COMMS][COMMA][SHIFTC][SHIFTK][3SPC]"
590 PRINT"[12SPC][SHIFT@][SHIFTL][5SPC]"
600 PRINT"[12SPC][SHIFTN][SHIFTM][5SPC]"
610 RETURN
```

CARTA





BLANCA

Tengo un micro ordenador Commodore 64 y el Floppy 1541.

Así mismo les ruego me proporcionen información sobre los programas EASY SCRIPT y CALL RESULT, y cualquier otro programa de proceso de datos y sobre idiomas, siempre refiriéndome para diskette.

JESUS UGARTE UGARTE,
LEON Y CASTILLO, 228 ATICO X,
LAS PALMAS-4

En este número publicaremos reseña sobre EASY SCRIPT.

Ver galería de Soft (página 2), en el próximo publicaremos un amplio artículo sobre el mismo. Tomamos nota sobre CALL RESULT.

Sería interesante que la revista no se tecnificase excesivamente pues se convertiría en incomprensible para la mayoría de los usuarios de Commodore que creo somos simples aficionados y no técnicos, también sería bueno para nosotros que se tocasen algunos POKES de direcciones de memoria en cada revista para que se fuese desvelando este verdadero misterio para los aficionados, lo que podía hacerse con pequeños programas que nos mostrasen su aplicación prácticamente.

Sería conveniente también que se explicasen paulatinamente los conceptos de buffer, puntero, vector, memoria intermedia de teclado..., etc.

Nada más, reciban un cordial saludo de su nuevo suscriptor.

Según podrás apreciar, intentamos dar un poco para todos, tanto para los novicios como para los muy veteranos e incluso para los niños (a partir del próximo nº 3).

Tomamos nota para comenzar en nuestro próximo con un glosario de términos y con "el misterio de los Pokes". De hecho nos gusta el término como título. Te lo robamos.

Con esto contestamos también a JOAQUIN LOZANO DE VALENCIA que nos pide un estudio de la diferencia del POKE entre el VIC y el 64. Así como a varios lectores que nos han llamado con la misma petición.

CONTRABANDO

Estimados amigos:

Me dirijo a vosotros porque en fecha no muy lejana voy a comprarme un cassette Commodore para poder grabar mis propios programas, he preguntado el precio pero nadie parece saberlo, por eso os escribo esta carta.

Ya que vosotros o ustedes lo fabricáis, deberíais saber el precio exacto en el mercado.

También desearía saber, si es posible el precio en Andorra. Muchas gracias.

Nota: Me ha gustado mucho la editorial del nº 1 de la revista Commodore World.

JULEN ARRUNTIA
C/ PIO XII, 13, Esc. B 2º C
PAMPLONA

El VIC vale 34.900, el C-64 79.900 y el Cassette 12.000. Hay una casa de Zaragoza B.M. que se anuncia aquí página 21. Llámales.

Oye majo..., eso de Andorra..., ¡no pretenderás que te ayude al contrabando!

RESPETO A LA PROPIEDAD, ¿CUANDO?

Queridos amigos:

Ante todo, felicitaros por esta nueva revista que viene a llenar un hueco muy importante para los usuarios de Commodore, ya que, a pesar de que la revista del Club era muy interesante, no cubría todo lo que ésta desde su primer número promete; tengo un 64 desde hace pocos meses y toda la información me parece poca, así que mi felicitación de nuevo y mis mejores deseos de éxito que no dudo tendréis.

Con mucho gusto me suscribiría pero el buzón de casa es lo bastante pequeño para hacerme desistir pues desaparecen o las estropean intentando meterlas, de todas formas os prometo ser un lector asiduo.

Os acompaño unas pequeñas experiencias mías que pueden encajar en la sección de MAGIA y en caso de que mereciera la pena publicarlas, os seguiría mandando.

DOMINGO MARQUEZ GALLEGO
PARQUE VOSA, 3, 4º C
MOSTOLES (MADRID)
TELEF. 614 21 48

Gracias por tus palabras. Pasamos tu receta a MAGIA. ¡Ah! y síguenos mandando cosas.

¿CLUB DE INFORMATICA?

Somos suscriptores suyos, y a la vez hemos organizado un Club de Informática, donde empleamos ordenadores Commodore 64 y Vic-20.

Les enviamos folleto del primer curso que hemos organizado y rogamos nos tengan al corriente de cuanta información consideren que nos sea de utilidad.

JAVIER GARCIA DIAZ

A través del folleto no estamos muy seguros si se trata de un Club o de una academia. Tampoco se dice cuál es la diferencia entre socios y no socios ni qué se necesita para ser socios. Ampliad esta información, por favor. Si sois Club os incluiremos en el próximo número en MARKET CLUB.

TODO DE TODO

Hola, amigos:

Os escribo atendiendo a la petición de sugerencias y opiniones que hacéis en el número 1 de Commodore World, y la impresión que he sacado después de leerlo supongo que es la misma de todos los demás suscriptores:

¡MAGNIFICA!

Me parece fenomenal, por ejemplo, que incluyáis cada mes un tema, como el de generación de caracteres, en el número 1, acompañado de un programa y explicado exhaustivamente, con ejemplos, ilustraciones, etc...

A este respecto, hay algunos temas que me gustaría ver desarrollados en próximos números, como puede ser una explicación detallada de las rutinas de KERNAL, su funcionamiento, utilidad, modo de introducirlas en un programa, etc., todo ello acompañado de un programa explicado línea a línea.

Otro tema que me gustaría dominar también es el de la composición de música con el conjunto de comandos del SUPEREXPANDER; podéis explicar, por ejemplo, cómo pasar de la música escrita en pentagrama con corcheas y demás símbolos musicales a la notación del cartucho. También sería interesante que de vez en cuando se incluyera en la revista una canción popular, un efecto de sonido, etc..., para usar con el cartucho, todo ello debidamente explicado.

Habréis observado que hago mucho hincapié en eso de explicar lo que decís en la revista, y es que mi experiencia en la etapa anterior de la revista era a veces de quedar defraudado cuando se tocaba un tema y tenía que usar esa información sin comprenderla (p. ej. cuando salió el programa en que aparecía BUG en TV, no entendía por qué tal variable tenía que tomar tal valor, etc...).

También me interesa mucho el problema de cómo borrar parte de un programa que sólo se ha de utilizar una vez en él, y así recuperar esta memoria para variables... Esto ya se tocó anteriormente, pero intenté aplicarlo a un programa mío, sin entender



cómo lo hacía, y no dio resultado, por eso os pido que no dejéis nada sin comentar, por sencillo que parezca, en lo sucesivo.

Pasando a otro orden de cosas, me gustaría mucho que se publicara un curso de programación en lenguaje máquina, partiendo de nivel cero, y que incluyera programas explicados, descripción de comandos, etc...

Os agradecería que informarais en la revista, o a mí personalmente, sobre la existencia o no de algún curso en lenguaje máquina editado en español, pues, que yo sepa, no existe ninguno, a pesar de la gran demanda que debe de haber.

Como ya he visto que en el número 2 vais a publicar un artículo sobre la creación de nuevos mandatos al VIC, no os pido nada por ahora en este sentido, pero si me gustaría saber cómo se hace para cambiar el comando LIST por LISTA, o READ por LEE, etc..., que tengo entendido que se puede hacer, y espero que lo publiquéis próximamente.

Siguiendo con la opinión que pedís en la editorial del número 1, os diré que me parece fenomenal que se incluyan varios juegos en cada revista, ya que un mayor nivel de profesionalización de la revista no debe estar reñido con la dedicación de parte de ella al esparcimiento y la distracción.

Una buena idea por vuestra parte ha sido la de hacer esos sorteos de software periódicamente, creo que esto potenciará más la participación de los usuarios.

Bueno, por ahora no os voy a dar más el tostón, espero poder mandar alguna colaboración un día de estos. ¡Hasta pronto!

FRANCISCO SAEZ SOTO
C/ VIRGEN DEL ROSARIO, 28
ALCANTARILLA
MURCIA

¡Tú te quedas solo pidiendo! Muchas cartas así y ya tenemos ideas editoriales para todo el año.

Me temo que nosotros tampoco conocemos nada editado aquí sobre lenguaje máquina.

Los otros puntos que mencionas se irán tocando a lo largo del año.

Mientras tanto, incluyo tu dirección por si algún lector quiere cartearse contigo sobre estos temas.

Paso tus recetas a MAGIA.

¡Ah! y gracias por los piropos.

En los números 11, 12 y 13 de Club Commodore salieron unos artículos sobre este tema. Pretendemos comenzar un curso paso a paso en el próximo número. Asimismo apareció un mapa de memoria para el Vic 20 en el nº 1 en el que faltaban las siguientes posiciones:

$030C 780 almacenamiento temporal de A durante SYS.

$030D 781 almacenamiento temporal de X durante SYS.

$030E 782 almacenamiento temporal de Y durante SYS.

$0304 783 almacenamiento temporal de F durante SYS.

En el nº 7 apareció el mapa de memoria para el C-64. En el nº 15 referente a las tablas de KERNAL apareció un artículo con unas tablas. En el futuro (no prometo en qué número) trataremos este tema más a fondo con ejemplos prácticos.

NUEVO CLUB

Hace poco tiempo quedó establecido y reglamentado el Club de Programación Alaiz, en Pamplona. Está dirigido a estudiantes de B.U.P. y C.O.U. de esta ciudad, y el fin que tiene es la introducción en el mundo de la informática y programación BASIC. No tiene, por lo demás, ningún fin lucrativo. El club propiamente posee equipos COMMODORE 64, VIC-20 y diversos periféricos (impresora, unidad de disco y cassette)

Así mismo, dado que en Pamplona los distribuidores no tienen una gran información de bibliografía Commodore, periféricos y Software, y dado que formamos un club de programación me gustaría nos enviasen amplia información en estos campos.

Las actividades que realizamos requieren esta información ya que dentro de los chicos que participan los hay ya adelantados en programación.

JESUS RAFAEL MARTINEZ

ESTAREMOS EN INFORMAT

Hola amigos de COMMODORE WORLD:

Me llamo Ramón y esta Navidad me regalaron un COMMODORE VIC-20, incluido el "DATASSETTE". Yo tengo una gran afición a los "Videojuegos" y cuando mi hermano me compró el "Micro-Sistemas" Nº 7, con el cual regalaban vuestro primer número, me paré al leer la editorial ya que en el apartado de "Futuro" había una cosa que me interesaba, las consultas, como decía me paré para escribir esta carta, puesto que me gustaría que me enviarais o publicarais en vuestra/nuestra revista información sobre todos los "Video-juegos" disponibles para el VIC-20, sean en cartucho o cassette.

Supongo que tendréis un "Stand" en el "INFORMAT-84", me pregunto si venderéis "Video-juegos", "Joysticks", "Paddles", cartuchos diversos, etc. Por favor contestadme lo antes posible, sea en la revista o por carta, adiós amigos.

Gracias por haber leído esta carta y hasta pronto.

RAMON PLANS ESPERABE
C/ CORTS CATALANES, Nº 6, 3º-4.ª
CALDES DE MONTBUI
(BARNA)

Bienvenido a la familia Commodoriana, Ramón. Estaremos en Informat-84 por supuesto, pero compartiendo el "stand" bien con la editorial de MicroSistemas o con Microelectrónica y Control, todavía no está decidido. Nosotros personalmente no venderemos Soft excepto las cintas de programas de la revista, pero seguro que habrá bastantes distribuidores de productos Commodore que muy probablemente lo anunciarán en nuestras páginas de abril. Repito tu petición sobre soft en la columna correspondiente.

ESTIMADOS AMIGOS:

Por la presente os remito el "BOLETIN DE SUSCRIPCION" para la "nueva" Revista y unas dudas por mi parte. En el mes de julio pasado comenzó mi suscripción a la Revista "CLUB COMMODORE" y por lo tanto he recibido 6 números de dicha revista y el primer número de esta nueva, por lo que aún me quedan 4 números para acabar con la antigua suscripción, pero no he visto que hagáis ninguna referencia a la situación en la que quedamos los "antiguos" suscriptores, es decir, ¿recibiremos la nueva revisa igualmente sin mandar la nueva suscripción hasta que la anterior finalice? En este caso el nuevo Boletín que os adjunto podéis guardarlo hasta que acabe mi suscripción o remitirmelo de nuevo.

Me gustaría me permitierais que haga algún comentario respecto a los artículos y notas a los programas que han aparecido en "CLUB COMMODORE" y en "COMMODORE WORLD".

Creo que en algunos de éstos se dan largas explicaciones, algo muy interesante, pero esto ocurre casi siempre con las líneas de programa en "Basic" o programas escritos en este lenguaje. En cambio cuando se refiere a algún programa en código máquina las referencias son mínimas y las explicaciones son del todo un jeroglífico para mí, puesto que aún no he encontrado ningún manual que explique con detalle, como podría serlo para el "Basic", cómo utilizar el código máquina, por qué y cuándo. Sería de desear por mi parte hubiera una sección en la nueva revista tal como "CLAVE" en la que se enseñe o se explique de alguna manera cómo programar en código máquina.

Y para finalizar esta, demasiado larga, carta, felicitéis de mi parte a Don Alfonso Izquierdo Font por su excelente y sencillo programa "DIBUVIC" que, aunque un poco corto de explicaciones para poder sacarle el mayor provecho, para los que no entendemos de Geometría, resulta muy interesante.

FELICIDADES Y MUCHA SUERTE A TODO EL "EQUIPO" de COMMODORE WORLD.

RAMON RONCERO SANCHON
C/ JACINTO VERDAGUER, 36
MADRID-19

¡Gracias!

Aprovechamos tu carta para avisar a todos los "viejos" suscriptores que andan un poco despistados como tú. Todo los antiguos suscriptores pasan automáticamente a ser suscriptores de Commodore World hasta que finalice su tiempo de suscripción. Si alguno nos ha enviado una suscripción duplicada que nos lo comunique, por favor para devolverla o dejarla en depósito para renovación, a elegir.

Sobre lenguaje máquina te digo que lo mismo que a Francisco Sáez. Nos meteremos con ello lo más pronto posible.

Alfonso Izquierdo, un suscriptor, recoge tu felicitación desde estas páginas.

Pasamos tu pregunta a SEAMOS PREGUNTONES.

SEAMOS PREGUNTONES



Me gustaría me contestaseis al siguiente problema: Tengo un VIC-20 al que amplío la memoria con un cartucho de 8K RAM y un SUPER-EXPANDER colocados en el "VIC-20 Expander" de INDESCOMP (de hasta cuatro cartuchos), pero ocurre lo siguiente: 1.—En el caso de conectar los dos cartuchos la memoria RAM libre final que aparece en pantalla no es el resultado de la suma de las 8K RAM el SUPER EXPANDER y el propio equipo base. 2.—Si conectados los dos cartuchos y cualquiera de ellos se desconecta mediante el interruptor del "Expander" o lo quito de este (siempre con la alimentación desconectada) el resultado una vez vuelta a conectar la alimentación, es que una vez aparecido en la pantalla la "presentación", automáticamente desaparece como si se hubiese tecleado RUN/STOP y RESTORE y cada vez que se escribe algo, ocurre lo mismo. 3.—Sucede que en programas largos en los que se utiliza el modo GRAPHIC 2 aparece OUT of MEMORY pero consultando FRE(x) todavía queda memoria y si se elimina la instrucción GRAPHIC 2, el programa se ejecuta hasta el final, pero claro, sin alta resolución.

¿QUE HACER?

RAMON RONCEROS SANCHON
C/ JACINTO VERDAGUER, 36
MADRID-19

1º Cuando se utilizan SKRAM (o más) con el Super-Expander, los 3K RAM de este último están disponibles pero no para BASIC, sólo para lenguaje máquina.

2º Comprueba que estén bien introducidos los cartuchos en sus conectores.

El fallo que dices puede ocurrir si el expansor no dispone de resistencia de "Pull-up" asociadas a los interruptores (¿Fallo de diseño...?), pero no puedo asegurarlo ya que Indescomp no nos ha enviado ninguna muestra de sus productos.

3º El error "OUT of Memory" puede ser debido a la existencia de algún "GOSUB" sin "RETURN". Aunque disponga de memoria suficiente el stack puede llenarse dando lugar a este error.

Prueba este programa:

```
10 GOSUB 10
```

y consulta: FRE(X)

ATENCION DISTRIBUIDORES

Los siguientes lectores nos piden información sobre soft y productos Commodore.

Software - Educación y Negocios, José Antonio Herrera Arranz, Bristol Mayer, Isla de Java, 1, Teléfonos: 729 48 88 (Despacho) - 705 52 49

Me gustaría que me enviarais o publicarais en vuestra/nuestra revista información sobre todos los "Videojuegos" disponibles para el VIC-20, sean en cartucho o cassette.

Ramón Plans Esperabe. C/ Corts Catalanes, nº 6, 3º-4.ª. Caldes de Montbui (Barna).

Así mismo, dado que en Pamplona los distribuidors no tienen una gran información de bibliografía Commodore, periféricos y Software, y dado que formamos un club de programación me gustaría nos enviasen amplia información en estos campos.

Las actividades que realizamos requieren esta información ya que dentro de los chicos que participan los hay ya adelantados en programación.

Club Alaiz. Plaza Monasterio Santa Gemma, s/n. Teléfonos: 25 44 80 - 25 77 04. Pamplona.

Más información de otras publicaciones y programas establecidos.

Tengo un C-64

Juan López Ferrer, Plaza de Careaga, 9, 3º C. Teléfono: (951) 23 26 09. Almería.

Agradecería información sobre la existencia de cartuchos en el mercado para el C-64.

Jesús J. Inserte Peralta. C/ Columbano, 4. Valencia-20.

Me interesaría si ello fuera posible el que me remitierais un catálogo/lista de precios de los posibles accesorios o complementos que se pueden disponer para un VIC-20. También bibliografía en España traducida.

José Luis Lozano. Rambla Justo Oliveras, 81, 4º, 1.ª. Hospitalet de Llobregat. Barcelona.

Les rogamos que si tienen algún libro sobre programas, nos lo comuniquen, tipo de programas, precio y demás.

Ismael Iglesias Filgueira. Dirección de envío: Calle Arco, 9 y 11 - 2º E. La Coruña-3.

Os rogaría me enviarais una lista de publicaciones (libros, revistas, etc...) en castellano, sobre el COMMODORE 64.

Román Elizalde. Eichstrabe, 46. 3000 Hannover-1

Nos da casi vergüenza tener que admitir que estamos casi en las mismas condiciones que vosotros. Desde aquí hacemos una llamada a todos los distribuidores para que nos envíen listas de soft y productos disponibles con precios.

Desde ya, podéis poneros en contacto con las firmas que se anuncian en la revista. Ellas os ayudarán sin reservas.

Al publicar vuestra dirección los distribuidores pueden enviaros también información directamente y, por favor remitídnosla en bien de todos.

DISTRIBUIDORES

Si queréis vender vuestros productos no hagáis un misterio de ellos. "El buen paño en arca se vende", pertenece a la prehistoria.


La MAGIA son trucos, la MAGIA es divertida.
La MAGIA es hacer lo que nadie se ha atrevido.
Magia
La MAGIA es una columna mensual llena de consejos, trucos, de esto y aquello del mundo del software,hardware y aplicaciones.
Cada mes, MAGIA les trae trucos breves y útiles de informática procedentes de todo el mundo - trucos descubiertos por los demás que hacen que la informática sea más fácil, más divertida o más animada.
MAGIA habla de ideas sencillas, programas de una sola línea, subrutinas útiles, hechos de informática poco conocidos y otras cosas de interés. Buscamos material nuevo o renovado que resulta ser de valor actual para usuarios de equipos Commodore y que puede utilizarse con un mínimo de tiempo, esfuerzo o conocimientos teóricos.
MAGIA resulta ser la fuente más completa de información para la informática práctica, además de ser un foro internacional para compartir trucos con otros aficionados. Envía tus trucos a:
COMMODORE WORLD
Pedro Muguruza, 4
Madrid-16
La revista "Commodore World" sorteará seis paquetes de software en julio y diciembre entre todas las contribuciones publicadas.



DETECTAR PULSACIONES

Cuando se utiliza una sentencia GET para detectar una pulsación de tecla, el hecho de que se hayan salvado las pulsaciones anteriores en el "keyboard buffer" (memoria tampón del teclado) podría constituir un problema. A menudo, cuando se termina el juego, el jugador tiene que volver a iniciar el juego mediante la pulsación de una tecla. A continuación, presentamos la forma habitual de hacerlo:

```
510 PRINT "PULSA TECLA PARA
    CORRER"
520 GETA$:IFA$=""THEN520
530 RUN
```

Estas líneas volverán a ejecutar el programa incluso si una tecla fue pulsada ***antes*** *de que se ejecutara la línea 510. El problema se solucionará si se añade la línea 500 FORI=-=1TO10:GETA$:NEXT.*

Se puede hacer lo mismo con **una sola** línea si las líneas 500-530 se sustituyen por la siguiente:

```
500 PRINT "PULSA CUALQUIER TECLA PARA
CORRER":POKE198,0:WAIT198, 1:RUN.
```

El comando "POKE198,0" borra el "keyboard buffer". "WAIT198,1" le indica al ordenador que espere hasta que se pulse una tecla.

Westmorelana Commodore Newsletter

COMILLAS

Cuando se utiliza la sentencia Print con la información entre comillas, muchas veces es aceptable omitir el segundo par de comillas. Por ejemplo, el ordenador tratrará las siguientes dos sentencias de la misma forma:

```
100 PRINT "MAGIC IS FUND"
110 PRINT "MAGIC IS FUND"
```

Si se elimina el segundo par de comillas, se ahorra un byte de memoria, una pulsación y un espacio en la línea de pantalla, lo cual puede ser importante a veces. Pero hay que tener cuidado la última letra de la información contenida entre comillas **debe** ser lo último de la línea del programa. Dada la presencia del comando GOTO en esta sentencia, el segundo comando, 120 PRINT "ABRACADABRA": GOTO 120, no puede ser eliminado.

Dada la presencia del punto y coma después de esta sentencia Print, hace falta el comando de cierre, 130 PRINT "LEGERDEMAIN"

L.F.S.

PUNTO Y COMA

Muchas veces se pueden eliminar los puntos y comas entre varias unidades de información que van a ser imprimidas en la misma línea. Si no existe ninguna duda sobre la posición final de una unidad de información y la posición inicial de la siguiente, los puntos y comas son innecesarios. En este ejemplo:

```
140 PRINT A$;B$;C$;D;"E"
```

una unidad de información se distingue perfectamente de otra mediante el uso de los signos de dólar y las comillas.

Esta línea puede ser abreviada de la siguiente forma:

```
140 PRINT A$B$C$D"E"
```

El punto y coma **debe** ser incluido en esta línea:

```
150 PRINT F;G
```

Si se suitara, el ordenador imprimiría innecesariamente el valor de la variable FG.

Jacinto Arroyo (Badajoz)

COMAS

Si se coloca una coma entre dos unidades de información en una sentencia Print, la segunda unidad se imprime en la siguiente posición de TAB previamente establecida en la pantalla. En el Commodore-64, existen cuatro posiciones de TAB por cada línea de pantalla, mientras que en el VIC-20, sólo hay dos. Las comas adicionales colocadas entre las unidades de información hacen que las posiciones adicionales de TAB se salten. La A y la B se imprimen en las posiciones TAB 1 y 4 de la siguiente forma:

```
100 PRINT A,,B
```

Pedro D'Asilva (Lisboa)

FORMACION DEL MARCO EN PANTALLA

Esta rutina imprime un marco alrededor de la pantalla del Commodore PET, pero también funciona para el C-64. Si se cambian unos cuantos números, también funcionará para el VIC-20.

```
3000 REM **RUTINA DEL MARCO**
3010 PRINT "(clear)";:FOR I=1TO39:
     PRINT F$;:NEXT:PRINT "(cur-
     sor up)"
3020 FOR I=1TO23:PRINT F$TAB
     (38)F$:NEXTI
3030 FOR I=1TO39:PRINT F$;:NEX-
     TI:PRINT "home"
```

En esta rutina, F$ puede representar cualquier carácter. El usuario puede sustituir F$ por su carácter preferido (entre comillas, por supuesto) o cambiar F$ cada vez que se dibuja el marco. Por ejemplo:

```
40 F$="X":GOSUB3000.
```

Pero es importante recordar que si esta rutina se utiliza como subrutina, hay que añadir una línea 3040 Return, y hace falta una sentencia End en algún sitio antes de la rutina para que el programa no entre en la rutina mientras éste se está ejecutando.

Mercedes Antón (Madrid)

CONTROL PROGRAMABLE DEL CURSOR

El cursor puede ser colocado en cualquier posición de pantalla mediante el uso de una rutina como la que se presenta a continuación:

```
10   X$="(39 crsr rights)":Y$="(24 crsr downs)
100  X=20:Y=10:GOSUB 3000
110  PRINT"SORCERY"
2999 END
3000 REM**POSICIONAR CURSOR**
3010 PRINT"(home)"LEFT$(X$,X)LEFT$(Y$,Y);:
     RETURN
```

Las variables X$ e Y$ deben de inicializarse al principio del programa y no volver a cambiarse. El código en la Línea 100 establece la posición deseada del cursor, y a continuación coloca el cursor en la columna X y la línea Y. (X=0 para la columna de la extrema izquierda, Y=0 para la línea superior). Al volver de la subrutina, la línea 110 impide la información deseada en dicha posición. La línea 2999 prohibe la ejecución no deseada de la subrutina.

Nuestro experto añade: Otro modo de posicionar el cursor es, POKE 781, línea:POKE 782, columna:POKE 783,0: SYS65520.

Begoña Garrido (Bilbao)

ADICIONAL CONTROL PROGRAMABLE DEL CURSOR

Una vez asimilada la técnica del truco anterior, se puede abreviar mucho mediante la eliminación de la línea 10 y la modificación de la línea 3010 para que quede de la siguiente forma:

```
3010 PRINT"(home)"LEFT$ ("(24 crsr downs)",Y)TAB
(X);:RETURN.
```

Ignacio Casanova (Tarragona)



CONTAR EN HEXADECIMAL



Si se cuenta en hexadecimal resulta más fácil acostumbrarse al sistema de numeración hexadecimal. El "signo de libra" o el "signo de números" indica que lo que sigue es un número, mientras que el signo de dólar constituye una costumbre ampliamente usada que indica la notación hexadecimal. También existe una costumbre para indicar la notación binaria, es decir el signo de porcentajes, pero no se utiliza con mucha frecuencia.

Javier Roman (Zaragoza)

TRUCOS DE UNA SOLA LINEA

Este truco de una sola línea convierte hexadecimal en decimal.

Convierte un número hexadecimal de cuatro dígitos, expresado como una variable de serie H$, en su equivalente decimal, expresado como la variable numérica D. Se presenta a continuación:

```
100 D=0:FORI=1T04:D%=ASC(H$):D%-48+(D%   64)*7:
H$=MID$(H$,2):D=16*D+D%:NEXT
```

Para ver cómo funciona la rutina, se añaden las dos siguientes líneas, y se ejecuta el programa:

```
50  INPUT"HEX";H$
150 PRINT D
```

Margaret Ittel (Washington)

OTRO TRUCO DE UNA SOLA LINEA

Este truco convierte el número decimal D en su equivalente hexadecimal H$ de cuatro dígitos:

```
200 H$="":D=D/4096:FORI=1T04:D%=D:H$=H$+CHR$
(48+D%-(D%>9)*7):D=16*(D-D%):NEXT
```

Este programa de conversión se puede comprobar si se añade, además de 250 PRINT H$, al programa en el truco anterior.

A.W. Grym (New York)

"POKES" DE COLORES

Es fácil recordar los valores de los "Poke" para los primeros ocho colores del VIC 64 - el valor del "Poke" es uno menos que el número que aparece en la tecla de color. BLK se encuentra en la tecla 1, así que el valor "Poke" es el 0; WHT se encuentra en la tecla 2, y el valor del "Poke" es el 1 etcétera.

L.F.S.

CARACTERES PROGRAMADOS

En el artículo sobre la creación de caracteres que aparece en el nº 1 de "Commodore World", se dice en la primera columna de la página 10 que se pierde la capacidad de utilizar el juego de caracteres pre-programado del VIC al crear caracteres nuevos en un programa y modificar los punteros para que apunten a la RAM (o así me lo parece, al menos), pues bien, la experiencia que yo tengo al respecto es la de estar un día probando un programa de juegos y empezar a hacer POKEs en la posición 36869 desde 255, que es el nº necesario para que el VIC leyera los caracteres programados del juego, hacia abajo; al llegar a 240, ví que los caracteres que tenía en pantalla pasaban de los programados a los normales del VIC, lo que comprobé después incluyendo este comando POKE 36869,240 en el programa y haciendo que un PRINT pusiera en pantalla letras cuyo código había modificado con anterioridad, lo que supongo que significará que de nuevo estaba leyendo en ROM.

Francisco Sáez Soto

REPETICION DE TECLA

"En el COMMODORE 64, sólo las teclas de movimiento de cursor, la de INST/DEL y la barra espaciadora presentan autorrepetición cuando se las mantiene pulsadas. Si se activa el bit 7 del registro 650, todo el teclado presentará esta característica. Para ello, basta teclear: POKE 650,128".

Rafael García Segura (Málaga)

HELICOPTERO

La que doy a continuación es una subrutina que tiene por misión simular el sonido de un helicóptero. Utiliza el comando SOUND, es decir, que sólo se puede ejecutar en presencia del SUPEREXPANDER. La variable JJ no tiene ningún efecto sobre el sonido, sólo determina el tiempo que está ejecutándose la subrutina.

```
10 FOR JJ=1 TO 20
20 FOR J = 60 TO 65
30 SOUND 128+J,0,0,0,15-INT (J/5)
40 SOUND 0,255-J,0,0,1+INT (J/5)
50 SOUND 0,0,128+J,0,15-INT(J/5)
60 SOUND 255-J,0,0,0,1+INT (J/5)
70 NEXT J
80 NEXT JJ
90 SOUND 0,0,0,0,0
```

Francisco Sáez Soto

FUNCIONES ESPECIALES DEL TECLADO

Introduciendo directamente o en programa —POKE 650,255— se consigue que todo el teclado se vuelva repetitivo. —POKE 650,0— lo retorna a la normalidad.

Pulsando la tecla Control y al mismo tiempo S, pone el cursor al principio de la pantalla. Pulsando Control T: Delete. Control Q: Cursor abajo. Control ; : Cursor derecha. Control =: Pone el cursor del 1. fondo. Control E: Lo retorna a su color. Control N: Activa minúsculas. Control H: Fija Mayúsculas o minúsculas. Control I: Suelta mayúsculas o minúsculas. Control M: Return.

Esto se puede utilizar directamente o en programa. Aunque son repetición de las funciones que conocemos, algunas veces pueden ser útiles. Pulsando Print"N control N sale N en inverso. Al ejecutar "Run" tendremos las minúsculas, es más cómodo que POKE o que CHR$ y más rápido.

Domingo Márquez

COMO SIMULAR LA INSTRUCCION "PRINT AT"

Muchos usuarios del VIC-20 y del COMMODORE 64 echan de menos la instrucción
PRINT AT (coordenada vertical), (coordenada horizontal) que otras versiones de BASIC sí poseen. Esta instrucción permite posicionar el cursor en cualquier punto de la pantalla, tras la mera indicación de la fila y la columna en que deseamos situarlo. Una pequeña subrutina de una sola línea logra incorporar esta facilidad a cualquiera de los dos equipos arriba mencionados: basta con introducir el número de fila en la variable FILA, introducir el número de columna en la variable COL, y acudir a la subrutina que a continuación trascribo:

```
10000 POKE 781,FILA:POKE'782,COLUMNA:
POKE 783,0:SYS 6520:RETURN
```

Una vez incluida esta subrutina en nuestro programa, para colocar el cursor —por ejemplo—, en la columna 16 de la fila 9, escribiremos simplemente:

```
FILA=9: COL=16: GOSUB 10000
```

Mágico, ¿verdad?

Angel-Francisco Alegre Amor. C/ Dr. Marañón, 2, 6C, CACERES. Tfno.: (927) 24 98 85

```
### DIARIO ###

EMPRESA :CASA DE SOFTWARE, S.A.        DISCO : 1 FECHA :16-02-84   HOJA :   2

FEC. ASTO PT CONCEPTO      CUENTA DESCRIPCION                DEBE          HABER
------------------------------------------------------------------------------------
 ./.                                                         0.00           0.00
0101   1  1 APORTACIONES 570000 CAJA                    1000000.00
0101   1  2 APORTACIONES 100000 CAPITAL                                100000.00
0101   1  3 +IMP.A-1.P-2 100000 CAPITAL                                900000.00
0101   2  1 FOTOCOPIADOR 203000 MAQUINARIA               150000.00
0101   2  2 FOTOCOP.     570000 CAJA                                    50000.00
0101   2  3 LETRA ACEPT. 450000 EFECT.COMER.A PAGAR                    100000.00
0101   3  1 DE CAJA      572000 BANCO A                  900000.00
0101   3  2 ING. BANCO A 570000 CAJA                                   900000.00
0101   4  1 VARIOS       660000 MATERIAL DE               15000.00
0101   4  2 MAT.OFICINA  570000 CAJA                                    15000.00
0101   5  1 MOBILIARIO   208000 OTRO INMOV
0101   5  2 T/001        572000 BANCO A
0101   5  3 ACEPT.LETRA  450000 EFECT.COME
0101   5  4 ACEPT.LETRA  450000 EFECT.COM
0201   6  1 PRODUCTO A   600000 COMPRA A
0201   6  2 SU FRA       400000 PROVEEDO
0201   7  1 PRODUCTO A   600000 COMPRA A
0201   7  2 TRANSPORTE   650003 OTROS T
0201   7  3 SU FRA.      400000 PROVEED
0201   7  4 TRANSPORTES  570000 CAJA
0201   8  1 VENTA CONTAD 570000 CAJA
0201   8  2 PRODUCTO A   700000 VENTAS
0201   9  1 ORDENADOR    203000 MAQUI
0201   9  2 T/002        572000 BANCO
0201   9  3 ACEPT.LETRA  450000 EFECT
0201  10  1 PAPEL LIST.  660000 MATE
0201  10  2 PAPEL LIST.  570000 CAJ
1001  11  1 -IMP.A-8.P-1 570000 CAJ
1001  11  2 -IMP.A-8.P-2 700000 VEN
2901  12  1 ENERO        640000 ARR
2901  12  2 RBO.ALQUILER 572000 BA
------------------------------------------------------------------------------------
TOTALES
```

```
### CUENTA DE EXPLOTACION ###

EMPRESA :CASA DE SOFTWARE, S.A.       DISCO : 1 FECHA :16-02-84    HOJA :  1

CUENTA DESCRIPCION                              DEBE          HABER
--------------------------------------------------------------------
300000 COMERCIALES                              0.00           0.00
310000 PRODUCTOS TERMINADOS                     0.00           0.00
350000 MATERIAS PRIMAS Y AUX.                   0.00           0.00
600000 COMPRA A                            450000.00           0.00
601000 MATERIA A                                0.00           0.00
630000 I.T.E.                                   0.00           0.00
640000 ARRENDAMIENTOS                       20000.00           0.00
650003 OTROS TRANS. Y FLETES                 1000.00           0.00
660000 MATERIAL DE OFICINA                  20000.00           0.00
660001 COMUNICACIONES                           0.00           0.00
660002 PUBLICIDAD Y PROPAGANDA                  0.00           0.00
660005 VARIOS                                   0.00           0.00
       VENTAS A                                 0.00       50000.00
       ...ISIONES A                             0.00           0.00
                                                0.00           0.00
--------------------------------------------------------------------
                                                           441000.00
```

casa de software sa

c/aragón, 272 8º 6ª
tel. 215 69 52
barcelona-7

CONTABILIDAD 64

- Programa adaptado al Plan General Contable Español
- Cuentas de Mayor (3 dígitos) y de detalle (6 dígitos)
- 300 cuentas y 3000 apuntes por diskette
- Programa en cartucho y diskette simultáneamente
- Equipo: ordenador COMMODORE 64 ó SX-64, unidad de disco VC-1541, impresora VC-1525 ó MPS-801 y monitor ó T.V.

MENU

1 ASIENTOS, DIARIO Y CONSULTAS
  1 Entrada de asientos
  2 Diario
  3 Consulta de cuentas
2 EXTRACTOS Y SUMAS Y SALDOS
  1 Extractos de cuentas
  2 Balance de Sumas y Saldos
3 SITUACION Y CIERRE
  1 Balance de Situación
  2 Cuenta de Explotación
  3 Diario de Cierre
4 MANTENIMIENTO DE FICHEROS
  1 Parámetros
  2 Cuentas (altas-bajas-modificaciones)
  3 Asientos (bajas-modificaciones)
5 LISTADO DE FICHEROS
  1 Parámetros
  2 Cuentas
  3 Asientos
6 UTILITARIOS
  1 Copia de disco
  2 Borrado de periodo
  3 Vaciado de asientos
  4 Regeneración ficheros
  5 Regeneración/verificación disco
  6 Programación balance

Se incluye un completo **MANUAL DE UTILIZACION** así como un resumen del **PLAN GENERAL CONTABLE ESPAÑOL** y un capítulo con **NOCIONES FUNDAMENTALES DE CONTABILIDAD** para hacer posible la utilización de este programa a personas no expertas en contabilidad.

P.V.P. 24550.-

```
                                                         ...10
                                                         ...JA :   1
                                                             SALDO
                                                         000000.00
                                                        1000000.00
                                                         500000.00
                                                         300000.00
                                                         800000.00
                                                              0.00
                                                              0.00
                                                              0.00
                                                              0.00
                                                        -450000.00
                                                        -100000.00
                                                        -535000.00
                                                       -1085000.00
                                                         179000.00
                                                         665000.00
                                                         844000.00
                                                         450000.00
                                                              0.00
                                                              0.00
                                                          20000.00
                                                           1000.00
                                                          20000.00
                                                         491000.00
                                               50000.00       0.00
                                                              0.00
                                               50000.00  -50000.00
                                                              0.00
                                                              0.00
                                              3341000.00      0.00
```

```
### B...

EMPRESA :CASA DE SOFTWARE, S.A.

PASIVO
CAPITAL Y RESERVAS
   100 CAPITAL
DEUDAS CORTO
   400 PROVEEDORES
   450 EFECTOS CIALES.A PA
TOTAL PASIVO

EMPRESA :CASA DE SOFT
ACTIVO
INMOVILIZADO
MATERIAL
   203 MAQUINARIA
   208 OTRO INMOV
EXISTENCIAS
   300 COMERCIA
   310 PRODUCTO
   350 MATERIA
DEUDORES
   430 CLIEN
CTAS.FINANCI
   570 CAJA
   572 BANK
RESULTADOS
TOTAL ACT
```

```
                         FECHA :16-02-84     HOJA :  5
                              DEBE          HABER
                         3341000.00     3341000.00
.INADOS                        0.00
AS Y AUX.                      0.00
                                         450000.00
                                          20000.00
AS                                         1000.00
Y FLETES                                  20000.00
OFICINA                    50000.00
                                          50000.00
                          491000.00
                                         441000.00
GANANCIAS                 441000.00
-----------------------------------------------------
                         4323000.00     4323000.00
```

```
EMPRES
CUENT
-----
   310  PRO...
   350  MATERIAS P...
   400  PROVEEDORES
   430  CLIENTES
   450  EFECTOS CIALES.A PAGAR
   570  CAJA
   572  BANCOS                                 665000...
   600  COMPRAS MERCADERIAS                    450000.00
   601  COMPRAS MATERIAS PRIMAS                     0.00
   630  TRIBUTOS                                    0.00
   640  TRAB.SUMINIS.SERV.EXTER                 20000.00
   650  TRANSPORTES Y FLETES                     1000.00
   660  GASTOS DIVERSOS                         20000.00
   700  VENTAS MERCAD./PROD.TER                -50000.00
   701  COMISIONES                                  0.00
   730  INGRESOS ACCESORIOS                         0.00
   800  EXPLOTACION                                 0.00
   890  PERDIDAS Y GANANCIAS                        0.00
100000  CAPITAL                              -1000000.00
203000  MAQUINARIA                             500000.00
208000  OTRO INMOVILIZADO MAT.                 300000.00
300000  COMERCIALES                                 0.00
310000  PRODUCTOS TERMINADOS                        0.00
350000  MATERIAS PRIMAS Y AUX.                      0.00
400000  PROVEEDOR A                           -450000.00
400002  PROVEEDOR C                                 0.00
400004  PROVEEDOR E                                 0.00
400006  PROVEEDOR G                                 0.00
430000  CLIENTE A                             -100000.00
430001  CLIENTE B                                   0.00
430002  CLIENTE C                                   0.00
430003  CLIENTE D                                   0.00
450000  EFECT.COMER.A PAGAR                   -535000.00
570000  CAJA                                   179000.00
572000  BANCO A                                665000.00
600000  COMPRA A                               450000.00
601000  MATERIA A                                   0.00
```

```
### ASIENTOS ###

DISCO : 1 FECHA :16-02-84    HOJA :  1

       CONCEPTO          D/H          IMPORTE
----------------------------------------------
       APORTACIONES       D        1000000.00
       APORTACIONES       H         100000.00
       +IMP.A-1.P-2       H         900000.00
       FOTOCOPIADOR       D         150000.00
       FOTOCOP.           H          50000.00
       LETRA ACEPT.       H         100000.00
       DE CAJA            D         900000.00
       ING. BANCO A       H         900000.00
     1 VARIOS             D          15000.00
     2 MAT.OFICINA        H          15000.00
     1 MOBILIARIO         D         300000.00
     2 T/001              H          90000.00
     3 ACEPT.LETRA        H         105000.00
     4 ACEPT.LETRA        H         105000.00
     1 PRODUCTO A         D         250000.00
     2 SU FRA             H         250000.00
     1 PRODUCTO A         D         200000.00
     2 TRANSPORTE         D           1000.00
     3 SU FRA.            H         200000.00
     4 TRANSPORTES        H           1000.00
 8   1 VENTA CONTAD       D          55000.00
 8   2 PRODUCTO A         H          55000.00
 9   1 ORDENADOR          D         350000.00
 9   2 T/002              H         125000.00
 9   3 ACEPT.LETRA        H         225000.00
10   1 PAPEL LIST.        D           5000.00
10   2 PAPEL LIST.        H           5000.00
11   1 -IMP.A-8.P-1       D          -5000.00
11   2 -IMP.A-8.P-2       H          -5000.00
12   1 ENERO              D          20000.00
12   2 RBO.ALQUILER       H          20000.00
13   1 ...                D         100000.00
13   2 CAJA               H         100000.00
```



¡Y YO TAMBIEN CUENTO!



RINCON DEL 700



Software para el 700 (II)

El aspecto más importante del 700 en lo que a software se refiere es MEC/DOS, sistema operativo desarrollado por el Departamento de Software de Microelectrónica y Control, S. A. que incorpora 52 comandos más al Basic de Commodore para tratamiento de Ficheros (Relativos, Tablas y de Claves —Indexados—), gestión de Pantallas, Múltiple Precisión, Empaquetado y Desempaquetado de Campos, etc... En la VENTANA CBM de este número se describe en líneas generales lo que es MEC/DOS y lo que representa para el 700. Por ahora, seguiremos aquí hablando de software para el 700, aunque suponiendo siempre que este soft se confecciona con MEC/DOS.

La potencia que puede tener el 700 con MEC/DOS y un disco Winchester es sumamente considerable. Con diskettes, esta potencia está condicionada al modo de emplearlos (tipos y frecuencia de accesos), y a la capacidad de los mismos. No obstante, la disponibilidad de los ficheros de tipo Tabla (MEC/DOS) permite paliar esta posible desventaja que puedan ocasionar los diskettes en algunas ocasiones, operando en RAM los pequeños ficheros que pueden entorpecer el acceso a los grandes.

El secreto de una buena programación está en trabajar con el número de ficheros más apropiado en cada caso, y en el correcto diseño de los mismos. En los ficheros no pueden haber campos inútiles que entorpezcan el uso de los que realmente sí son necesarios. En los ordenadores, y más concretamente en los micros, el análisis debe esforzarse al máximo en su cometido para evitar que la máquina se convierta en un "muerto" que cause más problemas que ventajas a sus usuarios.

Todos sabemos lo importante que es el análisis en el desarrollo de las aplicaciones. El análisis es el punto intermedio resultante de lo que el usuario quiere que haga su ordenador (a veces verdaderos sueños imposibles) y lo que realmente puede hacerse en él. El analista es el intérprete entre el lenguaje profano en informática del usuario y la jerga propia de los programadores.

Pero más importante aún que el análisis es la filosofía general de trabajo de un analista, programador, o casa de software. En efecto, en lugar de estudiar cada aplicación por separado, es necesario tener unas normas u objetivos generales para que todos los paquetes tengan una estructura similar; ello repercute en claridad de análisis, rapidez de programación, y otras enormes ventajas que más adelante veremos.

En este artículo voy a comentar a nivel de análisis un paquete de gran divulgación. El tema es ya muy conocido, pero seguro que destaco algún aspecto interesante útil a alguien, o sugiero alguna idea nueva a algún analista.

El programa que casi todo usuario de ordenador tiene instalado es el de Contabilidad. Es el paquete más estandarizado que pueda haber (junto con el procesador de textos), pero también es cierto que es el que más variantes tiene. Básicamente, un programa contable debería hacer las siguientes tareas:

— Mantenimiento de Ficheros (Altas, Bajas y Modificaciones).
— Entrada de Apuntes y Corrección de los mismos.
— Listado Mayor.
— Listado Diario.
— Balances (Sumas y Saldos, Situación y Explotación).
— Cierre de Ejercicio.

Estos son los puntos básicos. Si un programa de Contabilidad no realiza alguno de ellos, es que ya ni siquiera es operativo.

A partir de aquí, los "softwareros" pueden añadirle todos los adornos que quieran para dar más importancia al paquete:

— Cartera de Efectos a Cobrar.
— Cartera de Efectos a Pagar.
— Control presupuestario (mensual o total).
— Estadística de Ratios.
— Listados alfabéticos de cualquier fichero.
— Confección de Balances atrasados (por mes).
— Programación manual de los Balances de Situación y Explotación, pudiendo operar con varios formatos.
— Diversos controles de cuadre.
— Captación de apuntes generados por otros programas.
— ...y un largo etcétera.

El problema se presenta cuando una empresa necesita un listado que su pro-

JORDI SASTRE



Logotipo de MEC/DOS tal como aparece en la pantalla de un CBM-B-700. (Diseño: J. Sastre. Foto: P. Masats).

CONTABILIDAD PERSONAL

MENU GENERAL

1. ENTRADA DE APUNTES.
2. LISTADO DE CUENTAS.
3. LISTADO DE INGRESOS Y GASTOS.
4. LISTADO BALANCE.
5. GRAFICOS DE BARRAS.
6. MANTENIMIENTO APUNTES.
7. MANTENIMIENTO CONCEPTOS.
8. MANTENIMIENTO CUENTAS.
9. CIERRE DE EJERCICIO.
10. SEGURIDAD.
0. FIN DE PROGRAMA.

NUMERO DE PROGRAMA?

Aspecto de la pantalla "Menú General" de un módulo de contabilidad personal realizado con MEC/DOS en un CBM-B-700. (Diseño: J. Sastre. Foto: P. Masats).

grama no es capaz de hacer, y cuando una empresa tiene que soportar todas las funciones extras que su programa hace, pero que en realidad no le hacen falta. En el primer caso el ordenador está infrautilizado (mal programado) o es demasiado pequeño; mientras que en el segundo está mal instalado: si se suprimieran del programa todos los procesos que el usuario no utiliza, seguro que los tiempos de ejecución bajarían bastante y la capacidad de almacenamiento para apuntes y cuentas se incrementaría considerablemente.

Este problema tiene varias soluciones, algunas más factibles que otras. Las casas de software podrían tener una amplia gama de programas de Contabilidad, cada uno enfocado a un tipo de empresa diferente. Ello exigiría un notable esfuerzo de la misma en el mantenimiento de tales programas. Pueden tener un programa-base al que implementarían todas las opciones que el usuario deseara, como si hicieran los programas de Contabilidad "a medida". El inconveniente es el mismo que tienen todas las aplicaciones "a medida": el tiempo de análisis, de programación, de pruebas, de depuración, etcétera, implica un alto coste del programa. Además que se encontrarían con que cada cliente tiene un programa diferente, a saber cuál es el mejor. Una solución más factible es tener el programa repartido en módulos interactivos, que el usuario elegiría a su gusto. Este es el sistema más deseable, pero el más difícil de conseguir. Podría estar compuesto de:

— Programa base (puntos mencionados al principio).
— Módulo de Cartera de Efectos.
— Módulo de Presupuestos.
— Módulo de Ratios.
— Módulo de Estadística Mensual.
— ...etc.

Existe también un caso, sobre todo en las personas que trabajan como profesionales liberales, en que ya el módulo base de contabilidad le viene grande. Fiscalmente no están obligados a llevar una contabilidad oficial, por lo que a ellos les bastaría controlar sus ingresos y gastos particulares. Tendrían suficiente con un programa que, además del mantenimiento de ficheros y entrada de apuntes, les confeccionara un detalle de ingresos y gastos, el listado de cuenta corriente del banco, y un balance que les informara de los beneficios. El escaso número de cuentas contables (básicamente cuentas de ingresos, gastos, caja y bancos) y de apuntes le redundaría en una mayor velocidad de ejecución y en la posibilidad de completar sus listados con diagramas de barras, estadística mensual, etc...

Actualmente, en nuestro país, por muy baratos que sean los microordenadores, no llegan a justificar su precio si se adquieren para llevar únicamente esta mini-contabilidad. Hay ordenadores personales, incluso familiares, capaces de gestionarla, introduciendo los apuntes después del combate intergaláctico y antes de la partida de comecocos. Pero un profesional de este tipo tiene otros trabajos que también son mecanizables: el abogado necesita un procesador de textos, el arquitecto un cálculo de estructuras, el constructor un seguimiento de obras, el que esté sujeto a la Estimación Objetiva Singular una relación de facturas emitidas, etc... A veces un ordenador es más útil cuando realiza las pequeñas tareas de cada uno, que mecanizando todo un complejo y voluminoso proceso.

Normalmente, el ordenador se compra para que realice el mayor número de tareas posible, con el mínimo presupuesto de maquinaria; es decir, mucho soft con poco hard. Si a eso le añadimos que, por definición popular infundada, el soft tiene que ser más barato que el hard, los distribuidores de ordenadores se las ven negras para confeccionar un soft barato que a la vez sea potente, cosas casi siempre incompatibles. Han recurrido a los estándares anunciando grandes programas de contabilidad, grandes programas de facturación y grandes programas de nóminas a un precio realmente barato. Han preferido hacer un gran paquete que lo pueda "hacer todo", antes que confeccionar cada una de las aplicaciones a medida. Ello repercute en los que realmente no necesitan tantos procesos y listados para mecanizar su empresa.

Hemos llegado al problema inicial planteado en este artículo: el programa estándard que no se adapta al usuario. Hasta que no arraigue entre el gran conjunto de consumidores de informática la idea de que la programación puede resultar más cara que el propio ordenador, el problema seguirá existiendo. Cuando ello ocurra, estos grandes paquetes podrán dividirse en otros más pequeños que se podrán modularizar entre sí. Por ejemplo, la gestión comercial de una empresa estará compuesta por:

— Módulo de Contabilidad (a su vez también modularizado).
— Módulo de Control de Almacén.
— Módulo de Control de Clientes.
— Módulo de Control de Proveedores.
— Módulo de Facturación (facturas, recibos y remesas).
— Y todos los módulos que hagan falta.

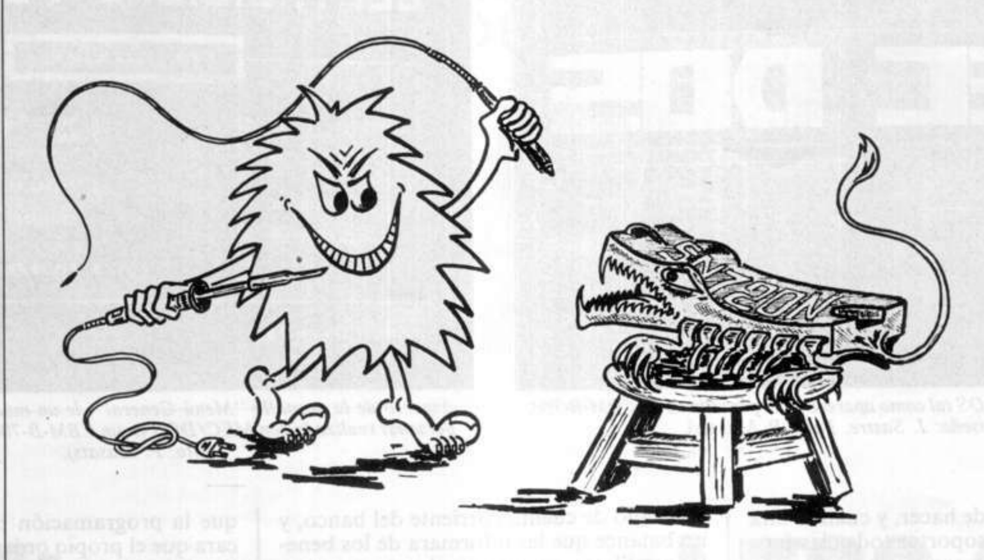
De esta manera, pueden haber muchos módulos sobre un mismo tema, por ejemplo el Control de Clientes: algunos controlarán las compras totales que hace cada cliente, otros lo controlarán mensualmente, habrán diferentes modalidades de pago pre-programadas, algunos módulos incorporarán el cálculo de rappels, otros tendrán estadística de compras por unidad, litro, kilo, etc... La elección apropiada de cada uno de los módulos que componen la programación permitirá obtener una aplicación casi a medida. Cuantos más módulos haya por elegir, más eficaz será la mecanización total.

Por ahora, los costes que implican confeccionar cada uno de estos módulos, que han de ser compatibles con los demás, aparte de poder funcionar por sí solos, es demasiado alto en el campo de los microordenadores. Estaremos más cerca de ello si empezamos a confeccionar pequeños paquetes, réplica de los grandes, que puedan comunicarse con algún otro pequeño paquete. El coste de la mecanización total de una oficina, a base de estos pequeños paquetes, puede resultar más caro, pero también más ventajoso al usuario, que dispondrá de programas a medida que no tienen etapa de depuración, pues están ya confeccionados. En realidad, cada uno de estos pequeños paquetes puede ser un módulo de una programación mayor.

Todo lo dicho no afecta únicamente a la serie 700 de Commodore, ni a los ordenadores Commodore en conjunto, pero ya estamos en lo de siempre: quien pega primero pega dos veces. ■



EL RINCON DEL VICCOLAGE



Los capítulos previos fueron publicados con anterioridad en COMMODORE CLUB. Aquellos lectores que los deseen, les rogamos que nos lo comuniquen para poder enviarles copia de estos capítulos.

El RS-232-C (II)

El RS-232-C en los ordenadores Commodore

Las últimas generaciones de ordenadores COMMODORE, (VIC 20, COMMODORE 64, CBM 700), incorporan de alguna manera el hardware necesario para efectuar la comunicación con otros periféricos u ordenadores mediante un canal RS-232-C. El VIC 20 y el COMMODORE 64 requieren un cartucho de interface, el VIC 1011A, que se conecta en el port del usuario de cualquiera de ellos. El CBM 700 incorpora de origen un port de comunicación serie RS-232-C.

El RS-232-C del COMMODORE 64 y del VIC 20 está dispuesto en el formato estandar de esta interface pero los niveles no son los adecuados del RS-232-C. El interface de COMMODORE (VIC 1011A) efectúa esta adaptación.

El acceso y programación del interface puede efectuarse desde BASIC o mediante ciertas rutinas del KERNAL.

El software para manejar la interface desde BASIC utiliza las siguientes instrucciones: OPEN, CLOSE, CMD, INPUT#, GET#, PRINT#, y la variable reservada ST.

Las instrucciones INPUT# y GET# toman los datos desde el buffer receptor y PRINT# y CMD colocan los datos en el buffer emisor.

Apertura de un canal RS-232-C

Sólo se puede abrir un canal de este tipo a la vez. Una segunda instrucción OPEN causa el reajuste de los punteros del buffer. Todos los caracteres de cualquiera de los dos buffers se perderán.

Se pueden colocar hasta 4 caracteres en el campo del nombre del fichero. Los dos primeros son caracteres de control y de comando. Los otros dos están reservados para sistemas futuros. La velocidad de transmisión, paridad y otras opciones se pueden seleccionar usando esta característica. Si se utiliza un carácter de control ilegal o que no está definido, el sistema envía los datos a una velocidad menor de 50 baudios.

Sintaxis en basic

La sintaxis utilizada para la apertura de un canal RS-232-C es la siguiente:

MANEL SANS



OPEN lfn,2,sa,
—REGISTRO DE CONTROL—
—REGISTRO DE COMANDO—

Logic filename —lfn.— El número de fichero lógico puede ser cualquier número entre 1 y 255. Si se utiliza un número superior a 128 se alimentará una línea después de cada retorno de carro.

Secondary address —sa.— La dirección secundaria en el VIC 20 y el COMMODORE 64 es recomendable que sea siempre un cero. En el CBM 700 existen varias opciones:

sa=0 - NO ACTUA
sa=1 - SOLO TRANSMISION
sa=2 - SOLO RECEPCION
sa=3 - TRANSMISION/RECEPCION
sa mayor que 128 - HABILITA LA CONVERSION CBM/ASCII

Registro de control

Es un carácter de un solo byte que contiene la información de velocidad de transmisión/recepción, longitud de palabra, número de bits de stop. Vea la figura 1.

Registro de comando

Es un carácter de un solo byte que define otros parámetros del terminal como son: Opciones de paridad, half/full duplex, modo de Handshake. Ver figura 2.

En un programa BASIC, la apertura de un canal RS-232-C debe hacerse antes de definir cualquier variable o tabla ya que automáticamente, al abrir un canal RS-232-C se efectúa un CLR. También se ha de tener en cuenta que son necesarios 512 bytes libres como mínimo antes de la apertura de un canal RS-232-C.

Para que quede claro el concepto de REGISTRO DE CONTROL y REGISTRO DE COMANDO pondremos un ejemplo. Supongamos que deseamos programar la interface como 7 bit de longitud de palabra, 1 bit de stop, 300 bauds de velocidad de recepción/transmisión, paridad inpar recepción/transmisión, full duplex y 3 líneas de comunicación.

El valor binario del registro de comando es 00100000 y su valor en decimal es 32.

Por lo tanto utilizando la sintaxis en BASIC debemos poner:
OPEN 2,2,0,chr$(38)+chr$(32)

BIT 6	BIT 5	LONGITUD DE LA PALABRA DE DATOS
0	0	8 BITS
0	1	7 BITS
1	0	6 BITS
1	1	5 BITS

3	2	1	0	VELOCIDAD (BAUDIOS)	
0	0	0	0	NO IMPLEMENTADO	
0	0	0	1	50 BAUDIO	
0	0	1	0	75	
0	0	1	1	110	
0	1	0	0	134.5	
0	1	0	1	150	
0	1	1	0	300	
0	1	1	1	600	
1	0	0	0	1200	
1	0	0	1	(1800)	2400
1	0	1	0	2400	
1	0	1	1	3600	[NI]
1	1	0	0	4800	[NI]
1	1	0	1	7200	[NI]
1	1	1	0	9600	[NI]
1	1	1	1	19200	[NI]

Figura 1. Estructura del Registro de Control (VIC-20 y C-64)

Recepción de datos desde un canal RS-232-C

En la recepción de datos de un canal RS-232-C, el buffer del VIC 20 y del COMMODORE 64 pueden almacenar hasta 255 caracteres antes de llenarse por completo. En el caso de que se intente almacenar más caracteres de los indicados se produce un OVERFLOW que es indicado en el bit correspondiente de la palabra de estado.

Si ocurre un desbordamiento de la capacidad del buffer, todos los caracteres recibidos desde que el buffer estaba completo se perderán.

Si desea recibir datos desde un periférico RS-232-C de gran velocidad deberá utilizar un programa en código máquina, ya que el BASIC es demasiado lento y probablemente perdería caracteres.

Bit núm.	7	6	5	4	3	2	1	0
1 bit stop	0							
7 bit datos		0	1					
No se usa				0				
300 bauds					0	1	1	0

El valor binario del registro de control es 00100110 y su valor en decimal es 38.

Bit núm.	7	6	5	4	3	2	1	0
Paridad impar R/T	0	0	1					
Full duplex				0				
No se usa					0	0	0	
3 líneas								0

Sintaxis en Basic

La sintaxis recomendada es:
GET lfn, —VARIABLE DE CADENA—

Si la palabra es menor a 8 bits, todos los bits no utilizados contendrán el valor cero.

Si un GET no encuentra datos en el buffer, devuelve una cadena vacía.

Si se usa INPUT, el sistema espera hasta que encuentra una cadena no vacía seguida de un retorno de carro. Por esto, si las señales CLEAR TO SEND o DATA SET READY desaparecen durante la ejecución del INPUT, la ejecución del programa se interrumpe. Por esto no se recomienda el uso de INPUT.



Envío de datos por el canal RS-232-C

El buffer de salida tiene una capacidad de 255 caracteres.

Sintaxis en Basic

Se puede utilizar:
CMDlfn
PRINT lfn, —LISTA DE VARIABLES—

No existe un retorno de carro en canal de salida. Esto significa que una impresora normal RS-232-C puede no imprimir bien a menos que se implemente algún tipo de HOLD-OFF (se pida al ordenador que espere), o exista algún buffer en la impresora. El HOLD-OFF se puede improvisar fácilmente en el programa. Si se trabaja en modo X-líneas el VIC 20 o el COMMODORE 64 efectúan el HOLD-OFF a partir de la señal CTS. En el COMMODORE 64 y el VIC 20 se han implementado las líneas RTS, CTS, DCD, como terminal de datos.

Cierre del canal RS-232-C

El cierre del canal RS-232-C descarta todos los datos presentes en los buffers en el momento de la ejecución (hayan sido o no transmitidos o impresos), detiene todas las transmisiones y recepciones del RS-232-C, ajusta la línea RTS y las líneas altas (Sout) de datos transmitidos, y destruye el contenido de los dos buffets del RS-232-C.

Sintaxis en Basic

CLOSElfn

Asegúrese de que se han transmitido todos los datos antes de cerrar el canal serie. Una forma de verificar esto desde BASIC puede ser:

```
100SS=ST:IF(SS=0 OR SS=8) THEN 100
110CLOSElfn
```

Variable ST o byte de estado.
(Ver fig. 3)

La variable reservada o variable de estado ST de longitud 1 byte, contiene toda la información necesaria para la validación de cualquier dato enviado o leído por el canal serie. Si los bits están a cero no se detecta error. La lectura de la palabra de estado provoca su borrado, y por tanto en los casos en que sean necesarias varias lecturas de la misma, se procederá a su asignación a otra variable. La variable de estado puede ser leída sólo cuando el canal serie ha sido el último en utilizarse en una operación de E/S.

7 6 5 4 3 2 1 0

OPCIONES DE PARIDAD — HANDSHAKE: 0-3 LINEAS, 1-X LINEAS

BIT 7	BIT 6	BIT 5	OPCIONES
-	-	0	PARIDAD INHABILITADA EN EMISION Y RECEPCION
0	0	1	PARIDAD INPAR RECEPCION/TRANSMISION
0	1	1	PARIDAD PAR RECEPCION/TRANSMISION
1	0	1	MARCA TRANSMITIDA PRUEBA DE PARIDAD INHABILITADA
1	1	1	ESPACIO TRANSMITIDO PRUEBA DE PARIDAD INHABILITADA

DUPLEX
0-FULL DUPLEX
1-HALF DUPLEX

NO SE USA
NO SE USA
NO SE USA

Figura 2. Estructura del Registro de Comando (VIC-20 y C-64).

Detalles de los conectores del canal RS-232-C

SEÑAL	VIC 20/C 64	VIC 1011/CBM 700	SENTIDO
Gnd	A,N	1,7	
R×D	B,C	3	IN
T×D	M	2	OUT
RTS	D	4	OUT
DTR	E	20	OUT
DCD	H	8	IN
DSR	L	6	IN
CTS	K	5	IN

Figura 3. Registro de Estado (ST) del RS-232.

```
[7] [6] [5] [4] [3] [2] [1] [0]
 :   :   :   :   :   :   :   :_PARITY ERROR BIT
 :   :   :   :   :   :   :_____FRAMING ERROR BIT
 :   :   :   :   :   :_________RECEIVER BUFFER OVERRUN BIT
 :   :   :   :   :_____________RECEIVER BUFFER—EMPTY
                               (USE TO TEST AFTER A GET#)
 :   :   :   :_________________CTS SIGNAL MISSING BIT
 :   :   :_____________________UNUSED BIT
 :   :_________________________DSR SIGNAL MISSING BIT
 :_____________________________BREAK DETECTED BIT
```



Una Excursión en Basic
Más allá del Manual
Capítulo II

Por Jeffrey Mills

En este capítulo vamos a descubrir cómo los valores se asignan a las variables — esto es un truco que cada programador debe tener en su repertorio.

En la Primera Parte de esta serie de artículos, empezamos a desarrollar un programa capaz de catalogar y listar los programas según un número de cinta.

Hablamos de los comandos preparatorios New, CLR y List. También hablamos de la numeración de las REM y las líneas. Hasta ahora, nuestro programa tiene el siguiente formato:

```
10 REM CATALOGO PROGRAMA/CINTA
20 REM ESCRITO POR: tu nombre
30 PRINT "(Shift-CLR/HOME)"
40 PRINT "CATALOGO CINTA"
50 PRINT
60 PRINT "101", "JUEGO 1"
70 PRINT "101", "JUEGO 2"
```

Ahora volvamos atrás para añadir las cabeceras de las columnas a nuestra lista. Utilizaremos la sentencia PRINT con comas. Se teclea:

```
52 PRINT "CINTA", "PROGRAMA"
54 PRINT "NO.", "NOMBRE"
56 PRINT
```

Hay que observar la forma en que las líneas se insertan entre los existentes números de línea. Esto constituye un buen ejemplo del motivo por el cual las líneas se numeran de diez en diez al comenzar un listado.

En vez de utilizar una distinta sentencia PRINT para cada programa de la lista, existe un método más sencillo. Sin embargo, antes de empezar a hablar de esto, hace falta saber un par de cosas más.

Más Terminología

1) Variable.—Una variable es una pequeña porción de la memoria del ordenador, a la cual se le asigna un valor, u otro contenido que podría cambiar, o variar, en el curso de un programa. Las variables podrían ser consideradas como pequeñas cajas o cajones de ficheros creados en la memoria mediante una sentencia del programa.

Existen varios tipos de variables. Para nuestros fines actuales, sólo hablaremos de las variables numéricas o de serie (letra o carácter).

A una variable en un programa se le puede asignar cualquier nombre dentro de las siguientes normas:

- Debe de empezar con una letra del alfabeto (A-Z), pero se puede utilizar tanto una letra como un número (0-9) en la segunda posición del nombre.
- Las dos primeras letras del nombre deben ser distintas de las dos primeras letras de cualquier otro nombre elegido. Estas dos letras son las que utiliza el ordenador para distinguir un cajón de fichero de otro. Por ejemplo, para el ordenador, BOY y BOAT son iguales, ya que sólo se miran las dos primeras letras.
- No se puede utilizar cualquier nombre de variable que empiece con las mismas dos letras que una Palabra Clave en Basic. (El Apéndice D del manual.)

Si la variable sólo va a contener números, se utiliza una variable numérica. Si va a contener caracteres (letras y símbolos), se utiliza una variable de serie.

Las normas para los nombres variables son las mismas que para las variables numéricas y la serie. Se le indica al ordenador que la variable va a contener letras y símbolos mediante la adición de un signo de dólar ($) al final del nombre.

Normalmente es mejor que los nombres variables sean lo más cortos posible, dado que cada carácter ocupa espacio en la memoria del ordenador. Sin embargo, si se llama a una variable que almacena un nombre NAME$, resultará fácil repasar el programa en el futuro para saber lo que representa dicha variable.

2) Sentencia de asignación.—Ahora que se sabe que el ordenador puede almacenar información en los cajones de ficheros llamados variables, hay que aprender la forma de introducir los valores en dichas variables. Esto se hace mediante una sentencia assignment (Asignación).

Supongamos que una variable se llama "A" y tiene que asumir el valor 5. Es suficiente teclear A=5. (También se puede teclear LET A=5, pero el comando LET se utiliza pocas veces, dado que el Commodore supone que se entiende LET si se teclea A=algún valor X.

Si se quiere cambiar el valor de la variable A a otra cosa, digamos 12, al teclear A=12, se borrará el 5, y el 12 se colocará en su lugar en el cajón de fichero A.

Leyendo los Datos

3) Read/Data.—El uso de los dos comandos Read y Data facilita la asignación de valores a las variables. Una explicación sobre estas dos sentencias Basic se presenta en las páginas 92 a 94 del manual.

La sentencia Data no es ejecutable. No es un verbo. El ordenador no hace nada cuando ve esta palabra en un programa.

Una sentencia Data simplemente almacena los números y palabras que serán utilizados por el programa. Se listan en el orden en que se utilizarán, y se separan mediante el uso de comas. A diferencia de la sentencia PRINT, estas comas no afectan en absoluto la salida de información en la sentencia Data. Su función es la de identificar la separación entre dos unidades de datos.

Cuando un programa se ejecuta, el ordenador recoge toda la información contenida en una sentencia Data y la almacena en una posición especial. De esta forma, la información puede ser recuperada en el momento en que la pide el programa. El usuario le indica al ordenador que recupere información de una sentencia Data, mediante el comando Read.

El comando Read recoge el siguiente valor de la lista, lo coloca en la variable especificada, y desplaza un puntero a la siguiente unidad de información de la lista. El puntero le indica a la siguiente sentencia Read dónde tiene que empezar.

En general, una vez que una unidad de información haya sido leída, no se vuelve a tener acceso a ella. El comando Read repasa la lista, leyendo toda la información, hasta el final de ésta. (Si el usuario intenta leer más información que la contenida en la lista de sentencia Data, el programa terminará con un mensaje de error de Out of Data [Faltan Datos]).

A continuación, se presenta una típica sentencia Data:

```
90 DATA 2,42,56,13
```

A continuación, se presenta una típica sentencia Read:

```
100 READ A
```

Mediante el uso del comando Read, el usuario le indica al ordenador que lea la siguiente unidad de información de la lista de sentencias Data y que coloque lo que lea en la variable especificada en la sentencia Read.

Cuando se ejecuta la línea mencionada anteriormente, A contendrá el número 2. Si la siguiente línea fuese 110 READ B, la variable B contendría el valor 42, y A seguiría conteniendo el valor 2.

Las sentencias Data también pueden contener datos alfabéticos. Por ejemplo:

```
90 DATA CAT, DOG, BIRD
```

Ahora la sentencia Read tendrá que especificar una variable de serie. Por ejemplo:

```
100 READ A$
```

Cuando se ejecuta esta sentencia, la variable llamada A$ (pronunciado A-string) contendrá la palabra "cat". Si la línea 110 se cambiara ahora a 110 READ B$, B$ contendría la palabra "dog", y A$ seguiría con su inquilino felino.

También se pueden mezclar datos numéricos y de serie (números y palabras) dentro de una sentencia Data. Por ejemplo:

```
90 DATA 1, GAME 1, 1GAME 2
```

Ahora las sentencias Read serian de la siguiente forma:

```
100 READ A
110 READ A$
120 READ B
130 READ B$
```

Después de ejecutarse estas sentencias, A contendrá el valor 1, A$ contendrá el "string" de caracteres GAME 1, B contendrá el valor 1 y B$ el "string" de caracteres GAME 2.

De la misma forma en que se utilizan las comas para separar las unidades de información dentro de una sentencia Data, se puede especificar que se lea más de una unidad de información en una sentencia Read. Por ejemplo:

```
100 READ A,A$,B,B$
```

Esta sentencia realiza lo mismo que las cuatro líneas anteriores.

Volviendo al Programa...

Dado que ya sabemos asignar valores a las variables, podemos imprimir el listado usando solamente una sentencia Print (después de las cabeceras). Sin embargo, hay otro concepto que debemos de tocar para poder realizar estas tareas de una forma eficaz. Este consiste en el bucle GOTO, del cual hablaremos con más detalle en otro capítulo.

De momento, es suficiente recordar que la sentencia GOTO le envía al ordenador a un número de líneas determinado dentro del programa. Si esto se hace al final de una serie de líneas, el ordenador vuelve a realizar el mismo juego de instrucciones una y otra vez, hasta que ocurre algo que produce un cambio. (La sentencia GOTO se explica en las páginas 30 y 123 del manual).

Vamos a sustituir las líneas 60 y 70 en nuestro programa Catalog por una sentencia Read y una sentencia Print. Se teclean las siguienes líneas:

```
60 READ N,P$
70 PRINT N,P$
```

Ahora se supone que nos hace falta una sentencia Data para acompañar la sentencia Read. Para que no resulte demasiado complicado, las sentencias Data tendrán asignados unos números de línea lo suficientemente altos como para permitir la inserción de otras líneas. Vamos a empezar con el número de líneas 9000. Se teclea:

```
9000 DATA 101,GAME 1,GAME 2
9010 DATA 102,GAME 3,GAME 4
9020 DATA 103,GAME 5,GAME 6
```

Las unidades de información contenidas en cada una de estas sentencias son un número de cinta, un nombre de juego, un número de cinta y un hombre de juego. Si se ejecuta el programa ahora, sólo se imprimirá el primer número de cinta y el primer nombre de juego. Aquí es donde entra en juego el bucle GOTO.

El programa entra en un bucle si se teclea 80 GOTO 60. Volverá a ejecutar las líneas 60 y 70, pero cuando termina la línea 80, regresará a la línea 60. Volverá a ejecutar las líneas 60 y 70 otra vez... y otra... y otra...

Cuando el programa haya leído toda la información contenida en la lista de sentencias Data, el programa termina imprimiendo lo siguiente en pantalla:

```
?OUT OF DATA ERROR EN 60
```

Existe una lista de todos los mensajes de error del Commodore en la página 150 y 151 del manual. Es normal que no signifiquen gran cosa para el principiante en estos momentos. Seguramente nos tropezaremos con muchos de ellos en nuestra pantalla a medida que vayamos escribiendo programas en Basic.

El Ejemplo 1 demuestra la salida del programa Catalog tal y como lo hemos desarrollado hasta ahora.

En el siguiente artículo hablaré de los métodos empleados para que el programa no termine en error, además de las formas de controlar la formación de bucles en el programa dentro del bucle GOTO. ■

Ficheros en disco (VI)

Indexados Secuenciales

Por Manuel AMADO



a) Introducción:

Este artículo pretende ser un puente entre la anterior serie de ficheros en disco CBM y otra serie que tratará sobre las estructuras de información en soportes magnéticos, o lo que es lo mismo, diversas estructuras de organización de los ficheros en las memorias de masa. El presente artículo es eminentemente práctico, dejando los rigores académicos para la presente serie, pues es simplemente un ejemplo de aplicación de las estructuras de ficheros ya conocidas (secuencial, relativo y acceso directo) que permite crear una estructura de fichero más compleja como es un fichero secuencial indexado.

Antes de empezar con la realización práctica de este fichero, veamos sucintamente qué es un fichero secuencial indexado. Con los distintos tipos de ficheros que nos proporcionan los discos CBM, podemos acceder a un registro determinado por dos métodos distintos, inherentes cada uno al tipo de fichero que se está tratando:

1.—Acceso secuencial:

Hay que leer todos los datos desde el principio del fichero hasta que se encuentra el dato o registro deseado. Propio de los ficheros secuenciales.

2.—Acceso directo:

Se accede directamente al registro deseado (ficheros de acceso directo y relativos).

Un fichero secuencial indexado (que a partir de ahora, y para abreviar, llamaremos ISAM —INDEXED SEQUENTIAL ACCESS METODE—) se llama indexado porque en él el acceso a un determinado registro se realiza mediante una clave de acceso. Esta clave puede ser un campo del registro, una parte de un campo, o una combinación de varios campos. Esta clave de acceso, que evidentemente lleva asociada la dirección en donde se halla el registro indexado por ella, hay que tenerla guardada en algún sitio. En el caso que nos ocupa, la clave se encuentra (generalmente) en una tabla ordenada, una detrás de otra. He aquí el motivo de que se llama secuencial, pues las claves de acceso están guardadas secuencialmente.

b) Diseño del fichero.

El ISAM que vamos a diseñar estará formado por dos ficheros distintos, el fichero índice que contiene las claves de acceso y el fichero indexado, en donde están los datos propiamente dichos. Veamos qué tipo de fichero de los que proporcionan los discos CBM se van a usar para el fichero índice y para el fichero indexado:

1. Fichero índice.

Para este fichero vamos a utilizar dos tipos de organizaciones diferentes, según el momento en que se vaya a procesar el fichero:

Acceso a los elementos del fichero índice: Para ello se va a usar una matriz bidimensional, en donde se va a guardar la clave en una dimensión y la dirección del registro asociada en la otra.

Almacenamiento en disco del fichero índice: Vamos a usar un fichero secuencial para guardar este índice en disco. La elección del fichero secuencial estriba en que solamente al principio del proceso se va a leer todo el fichero índice del disco, volcando su contenido en la matriz bidimensional, y al finalizar el proceso se va a volcar esta matriz al fichero en disco. Y el fichero más rápido en lectura y escritura es el secuencial.

2. Fichero indexado o de datos.

Este fichero tiene que permitir el acceso a un registro determinado. Los discos CBM nos proporcionan dos tipos de ficheros que cumplen este requisito, los ficheros de acceso directo y los relativos.

Para facilitar el comentario, vamos a centrarnos en primer lugar en el diseño de un ISAM con un fichero relativo como fichero indexado.



c. Diseño del programa.

El método de búsqueda de las claves en la matriz será el dicotómico, muy rápido en una tabla que esté ordenada.

Una vez fijados los principales tipos de datos a usar, vamos a ver el programa ejemplo de un ISAM con fichero relativo. Este programa se puede usar como rutina en un programa de gestión normal y contempla las opciones de mantenimiento de cualquier fichero, como son altas, bajas, modificaciones y consultas.

A continuación voy a describir el algoritmo del programa.

El programa tendrá dos cuerpos principales, el que gestionará el índice y el que recogerá los datos deseados del fichero indexado.

El programa se diseñará con un cuerpo principal de rutinas, comunes a los procesos principales de éste, que consistirán en altas, bajas, modificaciones y consultas.

El algoritmo es:

1) Inicialización de variables.
2) Rutinas principales.
3) Cuerpo principal.

3.1. Cargar el índice en la matriz.
3.2. Seleccionar operación.
3.3. Ejecutar operación (altas/bajas/-modificaciones/consultas).
3.4. Si se desea otra operación, ir a 3.3
3.5. Si no, grabar índice en disco.

3.3.1. If operación=altas then altas
If operación=bajas then bajas
If operación=consultas then consultas.
If operación=modificaciones then modificaciones.

—Altas:

(1) Entrar clave
Si clave="" then 3.2
Buscar clave. Guardar posición T
Si existe clave then entrar clave
Entrar resto del registro.
Pedir confirmación.
Si confirmación=no then entrar clave.
Buscar primer registro libre.
Insertar en índice clave+dirección primer lib. (posición T).
Incrementar número de registros ocupados.
Grabar registro.
Goto entrar clave, (1).

—Bajas:

(2) Entrar clave
Si clave="" then 3.2
Buscar clave. Guardar posición T.
Si no existe then entrar clave, (2).
Leer registro.
Visualizar registro.
Pedir confirmación.
Si confirmación=no then entrar clave.
Guardar número registro.
Borrar clave del índice.
Insertar clave falsa+número registro al final del índice.
Marcar registro como dado de baja.
Go to (2).

—Consultas.

(3) Entrar clave.
Si clave="" then 3.2
Buscar clave.
Si clave no existe then entrar clave.
Leer registro.
Visualizar registro.
Goto entrar clave, (3).

—Modificaciones.

(4) Entrar clave.
Si clave="" then 3.2
Buscar clave.
Si clave no existe then entrar clave.
Leer registro.
Visualizar registro.
Modificar registro (excepto campo clave).
Goto entrar clave, (4).

En el próximo artículo os entregaré el listado de uno de los muchos programas que podrían corresponder al algoritmo aquí descrito. No obstante, a ver si os animáis y mientras tanto hacéis vuestro propio programa según este algoritmo. ■

—Servicio gratuito para nuestros lectores particulares. Empresas 300 ptas., por línea.

MERCADILLO

• CBM 4.032. Intercambio programas. José Marcé Mestres. Calle Sevilla, 5. Tel. (93) 803 77 51, de 8 a 3, VILANOVA DEL CAMI (Barcelona). Referencia M-1.

• ACCESORIOS VIC-20: Ampliación de memoria 16K más varios programas, 13.000. Módulo de expansión para 6 cartuchos, 10.000. Cartucho lenguaje FORTH y manual, 7.000. Las tres cosas sólo por 25.000. Jaime. Tel. 245 46 56. BARCELONA. Referencia M-2.

• Desearía vender por 40.000 ptas.: VIC-20, con cartucho Super Expander, las dos partes del Curso, el "joy-stick" y un juego Indescomp, todo comprado en diciembre del 82. Fernando Martínez, calle La Roda, 39, 5º D. Tel. 23 41 82. ALBACETE. Referencia M-3. • Tengo un PET 2001/8K y desearía tener el cassette "Monitor para lenguaje Máquina". También desearía contactar con usuarios o clubs de PET si los hay. José Manual Cámara Mas, calle Castor, 32, bloque II, 3º, puerta 1. ALICANTE. Referencia M-4.

MARKETCLUB

• Se busca MODEM para Commodore 64. Precio a convenir. José M.ª Maci. Horno de Ladilla, 4, Montblanc, Tarragona 860063. Horas de comida. Referencia M-5

• Vendo VIC-20 (Comprado en noviembre 1983), 2 cursos de basic, libros, programas, 4 cartuchos, juegos en cassette, cassette, joysticks. Equipo ideal para iniciarse en informática. Todo por 59.000 (Valor real 90.000). Llamar de 9 a 11 h. noche. Joan Sanz. Tel. 3218064. Travesera de las Corts, 295. Barcelona, 29. Referencia M-6.

• Compraría Cassette y aplicación de memoria 8K para VIC 20. Precio a convenir. Manuel Aranda (4354900) oficina; 91 (6179331) casa. Referencia M-7.

CLUBS

Como presentación, valga decirles solamente que soy distribuidor de Commodore en Palma de Mallorca y al mismo tiempo aficionado a los microordenadores.

Estoy intentando coordinar los esfuerzos de unos cuantos clientes y amigos para formar un club de VIC y 64 en Palma de Mallorca. También os agradeceré si me podéis ayudar con una reseña en la revista para que los aficionados de Palma sepan que intentamos formar un grupo de amigos del VIC-20 y COMMODORE-64.
RAMON P. SERNA SOLER. FOTO ESTUDIO 2. Plaza de Sombrereros, 2. PALMA DE MALLORCA. Tel. 21 31 62. Referencia C-1.

Desea información sobre clubs en existencia y gente que quiera formar uno.
AGUSTIN QUEVEDO VELASCO.
OÑA, 55-4ºB. MADRID-34.
TEL.: 202 94 28
Referencia C-2.

Desea información sobre clubs en existencia y gente que quiera formar uno.
PABLO NISTAL ALONSO
c/EMPECINADO, 46 6ºB
MOSTOLES. TEL.: 645 21 70
Referencia C-3.

Hace poco tiempo quedó establecido y reglamentado el Club de Programación Alaiz, en Pamplona. Está dirigido a estudiantes de B.U.P. y C.O.U. de esta ciudad, y el fin que tiene es la introducción en el mundo de la informática y programación BASIC. No tiene, por lo demás, ningún fin lucrativo. El club propiamente posee equipos COMMODORE 64, VIC-20 y diversos periféricos (impresora, unidad de disco y cassette).
CLUB ALAIZ. Pza. Monasterio Santa Gemma, s/n. Tels.: 254480-257704 PAMPLONA. Referencia C-4.



BOLSA DE TRABAJO

PROGRAMADORES COMMODORE-64, excelente oportunidad para personas que dispongan de programas de gestión y utilidades y deseen comercializarlos en condiciones verdaderamente ventajosas (sólo programas de alta calidad). También programadores dispuestos a realizar programas a medida. Deberán ponerse en contacto con "EAF microgestión", preguntar por Francisco Arambudo, teléfono: (93) 231 95 87 de Barcelona, o escribir al apartado de correos 24143 de Barcelona. Referencia B-1.

SERVICIOS

• En Barcelona, clases de informática. PLAZAS LIMITADAS. Lenguaje BASIC. Prácticas con microordenador VIC-20. Prof. E. Martínez de Carvajal. Información: Tel. (93) 345 10 00. Señorita María José (mañanas) ó (93) 345 87 75. Sr. Martínez (fuera de horas de oficina). Referencia S-1.

• Programación de ordenadores personales; organización explotación de ficheros; programas ordenadores auxiliares, para cuestiones empresariales, profesionales, administrativas, científicas. Mora Mas. Carlos III, 41. Tel.: 339 98 29. BARCELONA-28. Referencia S-2.

• Tengo programa para confección de documentos, cartas, textos, etc. Permite escribir líneas reales de impresora visualizando por pantalla y avisando al final de cada línea por timbre. Visualización del texto entero; modificaciones de línea; insertar líneas; grabación disco o cassette; lectura de datos de disco o cassette; control por pantalla del texto a rectificar; acepta; espacios, etc.; imprime normal y doble ancho; pregunta número de copias; salto automático de página en impresora; margen izquierdo en la página. Amadeo Bargay. Pza. Hospital 5-5º. Tel.: 874 41 93. Manresa. BARCELONA. Referencia S-3.



GALERIA DE SOFT

NOVEDADES

Microelectrónica y Control ha sacado 3 nuevos programas para el C-64. Introducción al BASIC, parte I; EASY CALC RESULT (20.000 pts.) y MACRO ASSEMBLER (7.500 pts.). La reseña de estos dos últimos saldrá en el próximo número.

CURSO DE INTRODUCCION AL LENGUAJE BASIC, PARTE I

Con este curso no se incluyen cintas de programas, en su lugar, se ha añadido un apéndice con los listados de los programas a los que se refieren las unidades didácticas, para que el usuario los entre por sí mismo en el equipo y realice de esta manera una práctica de gran utilidad en BASIC.

Precio: 3.000 ptas.

Maneja tu ordenador

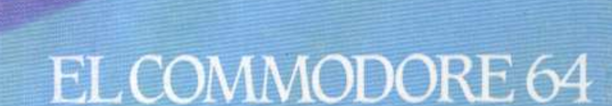
EN EL PROXIMO NUMERO...

las aventuras de ROMP y ROMO y d. Diodo

para los más jóvenes

ERRAZQUIN/84

El mejor ordenador personal del Mundo sólo cuesta 79.900 ptas.



1. Capacidad total de memoria RAM de 64 K. Interpretador BASIC extendido y sistema operativo residentes en ROM.

2. Dotado del más potente chip sintetizador de sonido diseñado hasta hoy, el COMMODORE 64 ofrece 3 voces totalmente independientes con una gama de 9 octavas. El programa puede controlar la envolvente, la afinación y la forma de onda de cada voz, convirtiendo al COMMODORE 64 en el mejor simulador de instrumentos.

3. Conectable directamente a toda una gama de periféricos, incluyendo unidad de discos, impresora de matriz de puntos o de margarita, plotter, comunicaciones locales y remotas..., y mucho más.

4. Pantalla de alta resolución en color con 320 × 200 puntos directamente direccionables. Capacidad en modo carácter de 25 líneas por 40 columnas.

5. El chip de video, único en su género, permite el uso de 8 «Sprites» (figuras móviles en alta resolución y color). Los «Sprites» pueden moverse independientemente por programa de «pixel» en «pixel».

6. Teclado profesional con mayúsculas y minúsculas, más 62 caracteres gráficos, todos ellos disponibles en el teclado y visualizables en 16 colores, en forma normal o bien en video invertido.

7. Encontrará a su disposición una completa gama de programas profesionales, incluyendo proceso de textos, sistemas de información, modelos financieros, contabilidad y muchas más aplicaciones.

8. Están en fase de desarrollo otros lenguajes tales como LOGO, COMAL, PILOT, etc.

9. Opción de un segundo procesador Z-80 para trabajar con sistema operativo CP/M (R).



MICROELECTRONICA Y CONTROL
c/ Taquígrafo Serra, 7, 5° Barcelona-29
c/ Princesa, 47, 3°, G Madrid-8

SLOGAN